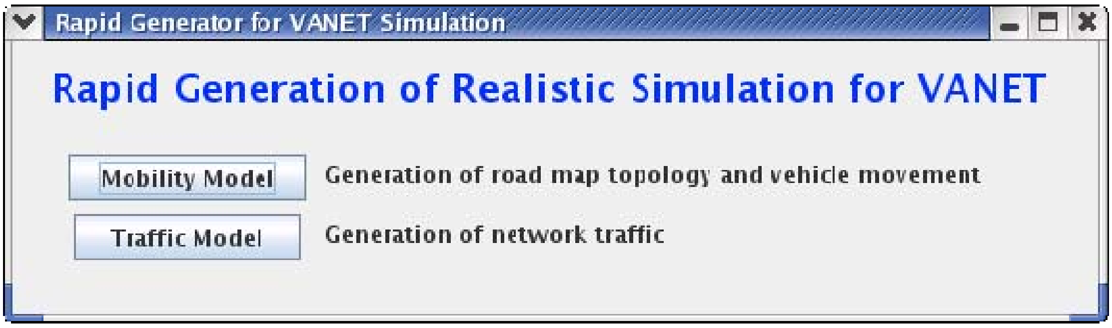
MOVE 使用手冊



Updated 30 July 2011

1. 事前準備

(1) 作業環境:

- Linux 或 Windows (XP/7)
- Java SDK 1.6 - http://java.sun.com
- SUMO version: 0.12.3 - http://sumo.sourceforge.net

在 Linux 系統下需要額外安裝:

- Xerces (XML-parser) - http://xerces.apache.org/xerces-c/index.html
- FOX-Toolkit (GUI Toolkit) - http://www.fox-toolkit.org/
- PROJ (Cartographic Projections Library) - http://www.remotesensing.org/proj/
- GDAL (Geospatial Data Abstraction Library) - http://www.remotesensing.org/gdal/
- NS2 version: 2.34 (all-in-one) - http://www.isi.edu/nsnam/ns/

備註：請參考各軟體在相關網站上的安裝細節，本文件只提供安裝特定版本的指令。

(2) 必要安裝的軟體套件及說明:

此範例針對在 Ubuntu 系統下，必須安裝額外的 SUMO 組件:

A. SUMO 部分

```
sudo apt-get update
```

SUMO wiki 有提供安裝細節:

http://sourceforge.net/apps/mediawiki/sumo/index.php?title=LinuxBuild

如果在 Windows 之下，可參考 SUMO wiki 中的安裝細節:

http://sourceforge.net/apps/mediawiki/sumo/index.php?title=WindowsBuild

B. NS2 部分

```
sudo apt-get install build-essential
sudo apt-get install tcl8.4 tcl8.4-dev tk8.4 tk8.4-dev
sduo apt-get install libxmu-dev libxmu-headers
```

安裝 Xerces:

參考網址: http://xerces.apache.org/xerces-c/build-winunix.html

1. 設定環境變數並指定 Xerces-C++的根目錄路徑

```
export XERCESCROOT=/home/temp/xerces-c-src_2_7_0
```

2. 設定組態檔

```
cd src/xercesc
./runConfigure -plinux -cgcc -xg++ -minmem -nsocket -tnative -rpthread
```

3. 建置 xercesc

```
gmake
make install
```

完成

安裝 FOX-Toolkit:

輸入下列指令進行安裝

```
./configure --with-opengl=yes --prefix=$HOME
make
make install
```

完成

安裝 PROJ 與 GDAL:

輸入下列指令進行安裝

```
./configure --prefix=$HOME
make
make install
```

完成

安裝 SUMO:

參考網址: http://sumo.sourceforge.net/wiki/index.php/LinuxBuild

輸入下列指令進行安裝

```
./configure --with-fox=$HOME --with-proj-gdal=$HOME --with-xerces=$HOME --prefix=$HOME
make
make install
```

完成

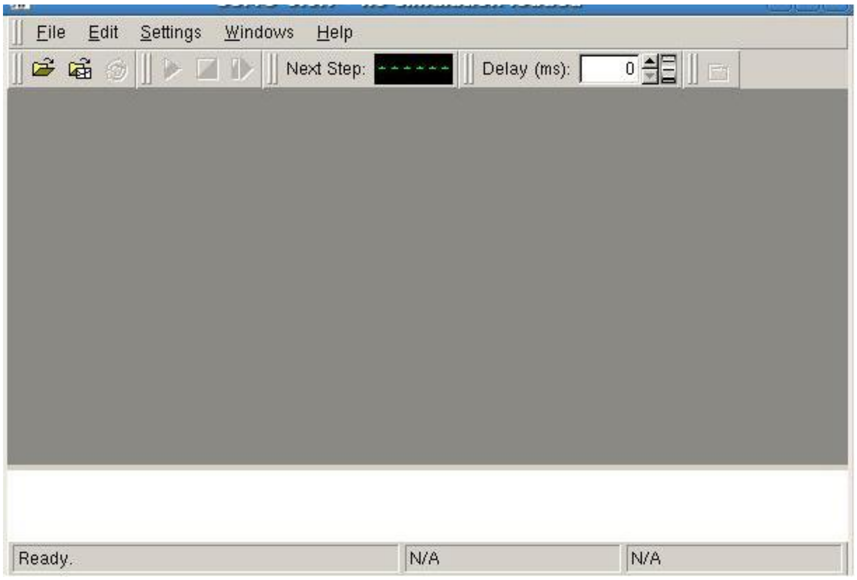
備註: 如果你遇到了 GCC 版本問題，可以嘗試使用下列指令解決

```
CC=gcc34
CXXFLAGS=-fpermissive ./configure –with-fox=$HOME –with-proj-gdal=$HOME –with-xerces=$HOME –prefix=$HOME
```

當安裝完成後，可以使用 GUI SUMO 來驗證安裝是否正確

```
cd /home/temp/sumo-0.9.8/src
./sumo-guisim
```

成功後將會看到下圖



安裝 NS-2:

下載位置 http://www.isi.edu/nsnam/ns/ns-build.html#allinone

完成下載 ns-allione-2.31.tar.gz 後，解壓縮

[root@localhost root]# **tar -zxvf ns-allinone-2.31.tar.gz**

安裝 NS-2

```
[root@localhost root]# cd ns-allinone-2.31
[root@localhost ns-allinone-2.31]# ./install
============================================================
* Testing for Cygwin environment
============================================================
Cygwin not detected, proceeding with regular install.
============================================================
* Build XGraph-12.1
============================================================
creating cache ./config.cache
checking for a BSD compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking whether make sets ${MAKE}... yes
checking for working aclocal
....…………………………………………………..
```

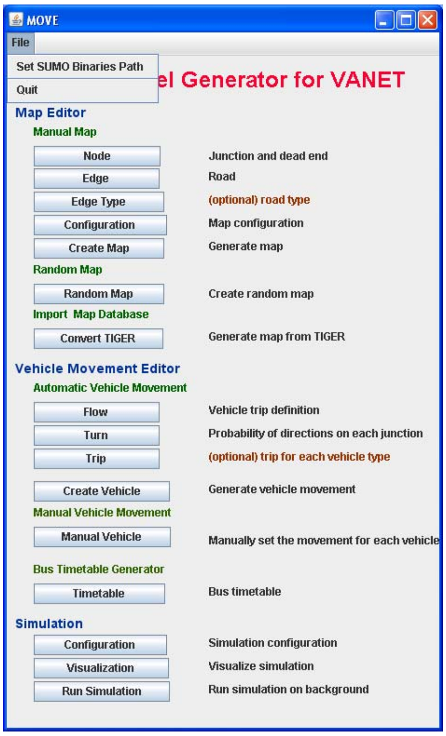
安裝完成

我們必須設定一些環境變數，並編輯.bashrc 檔的使用者主目錄，而且將額外設定以下變數:PATH，LD_LIBRARY_PATH，與 TCL_LIBRARY。

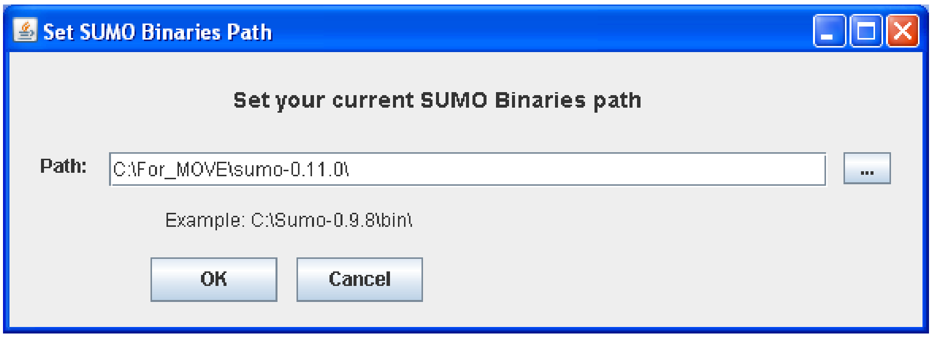
```
export NS_HOME=’pwd’/ns-allinone-2.31
export PATH= $NS_HOME /tcl8.4.14/unix: $NS_HOME /bin: $NS_HOME /tk8.4.14/unix:$PATH
export LD_LIBRARY_PATH=$NS_HOME /otcl-1.13: $NS_HOME /lib:$LD_LIBRARY_PATH
export TCL_LIBRARY=$NS_HOME /tcl8.4.14/library
```

建議可以修改~/.bash_profile 檔，如此一來就不必每次都重新登入和修改環境變數了。

設定 SUMO 的環境變數(只有在 Windows 版需要執行此動作)。



先選擇”Windows”，然後再執行“Set SUMO Binaries Path”，輸入 SUMO 的安裝路徑(如下圖)。例如我們的安裝路徑為“C:\For_MOVE\sumo-0.11.0\bin\”.



2. 安裝步驟及使用方式

(1) 安裝與執行:

I. 如果要直接編譯原始碼程式來執行 MOVE，使用指令編譯

```
javac *.java
java vanetsim
```

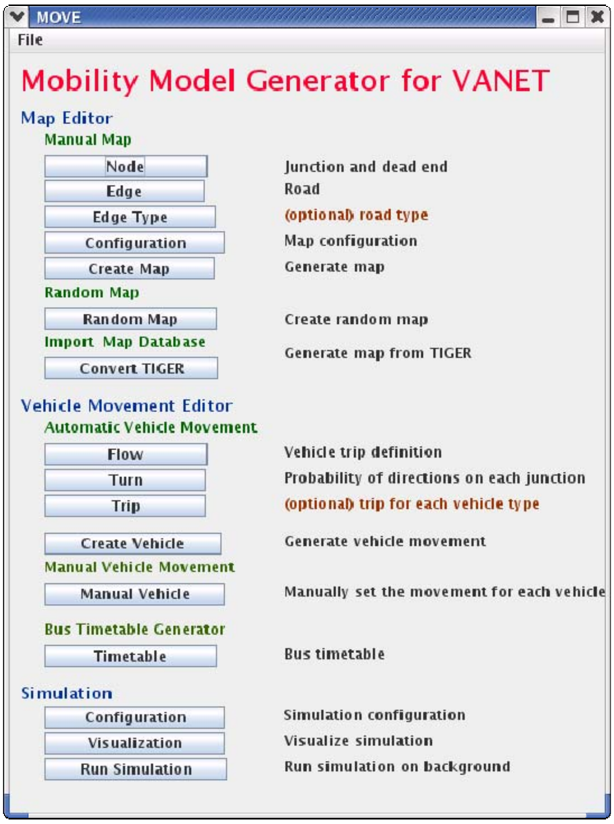
II. 或者以 jar 檔直接執行，輸入以下指令

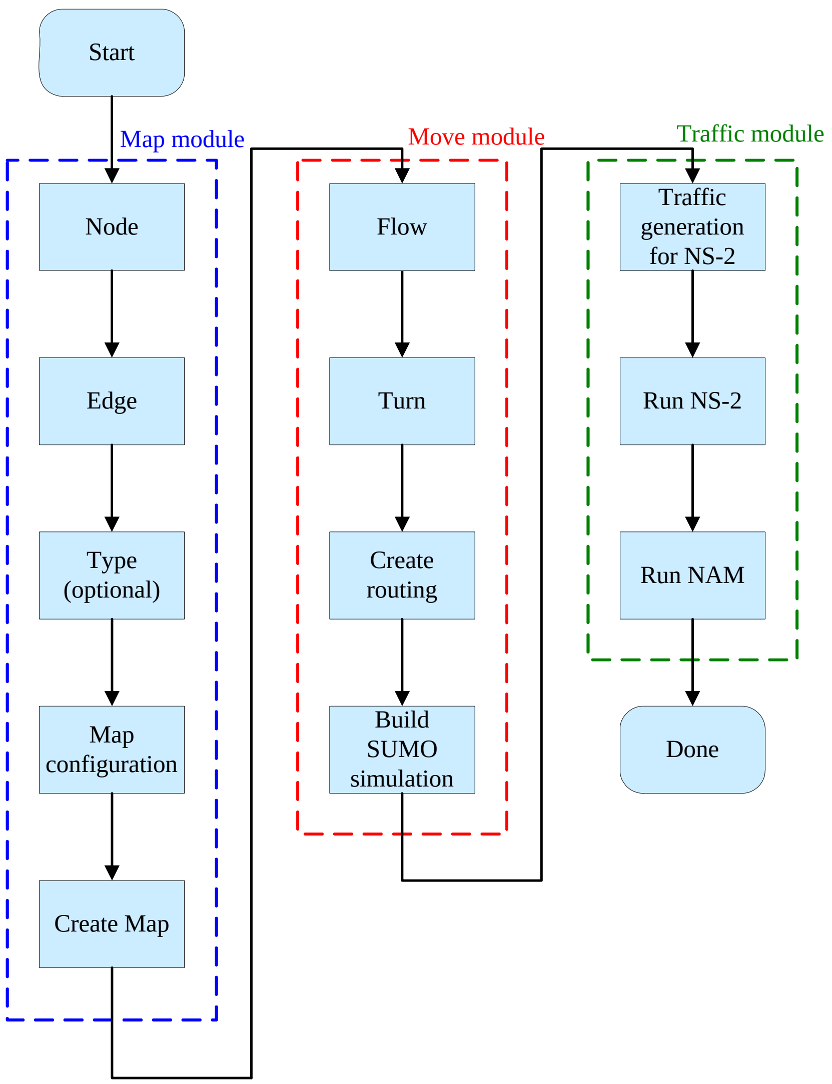
```
java -jar MOVE.jar
備註:若產生的移動路徑比較龐大，可以改為輸入
java –Xmx512m –jar MOVE.jar
```

(2) 功能使用

移動模型產生器(Mobility Generation)

軟體的此部分(MOVE-MObility model generator for VEhicular networks)可以經由 SUMO 產生 mobility model。首先在主選單上選擇 Mobility Mode 進入 mobility generation。





MOVE 的使用流程圖

我們將以圖 1 為情境範例，一步一步解釋每個功能。

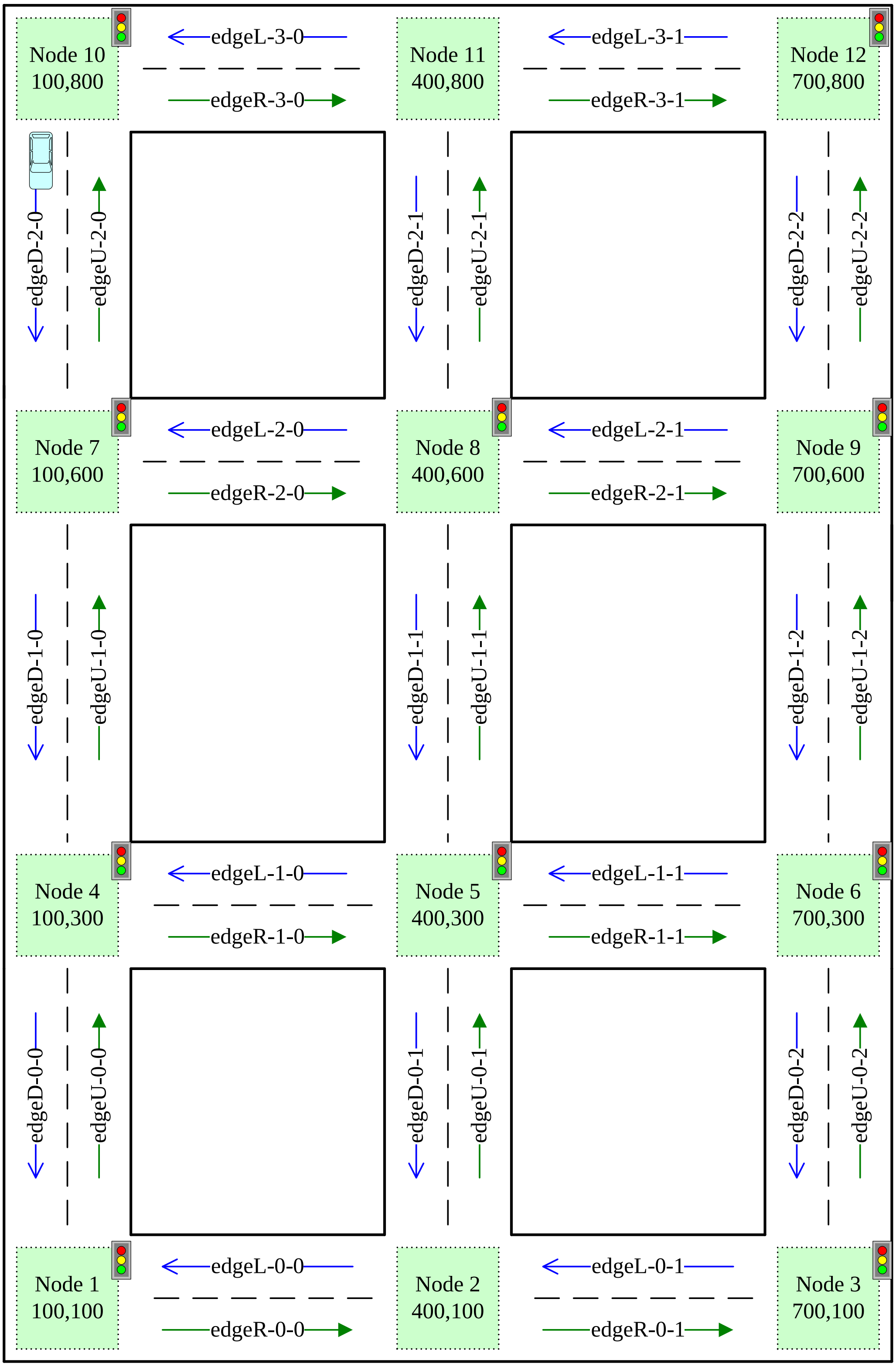
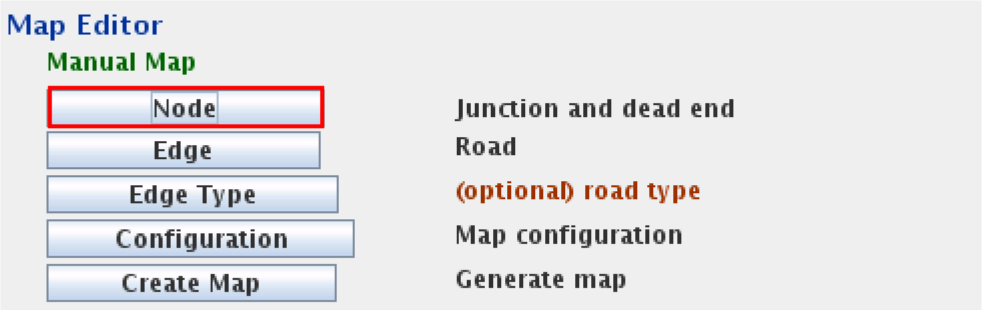


圖 1. 實驗範例

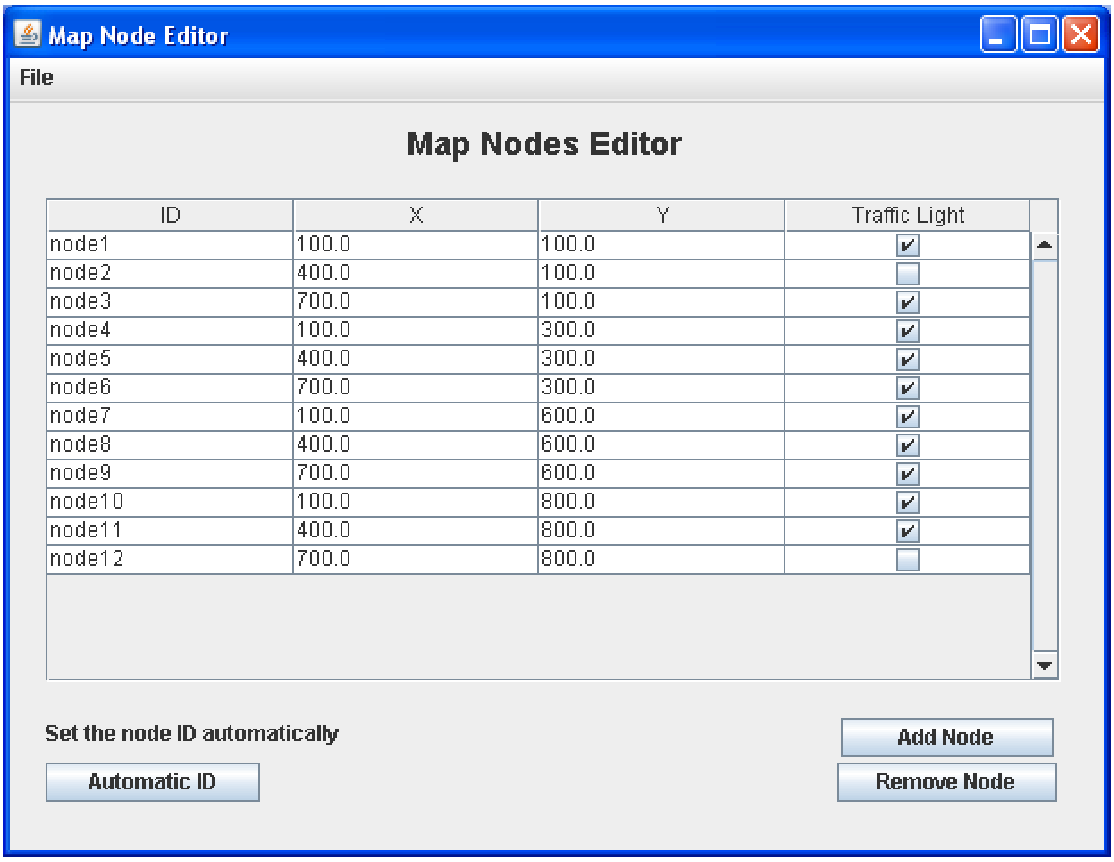
TAIWAN, NCKU university, CSIE, LENS lab.
Updater: Chine-Ming Chou

I. 地圖產生

步驟一. 手動設計地圖的十字路口 (範例檔名 ex_NODE.nod.xml)

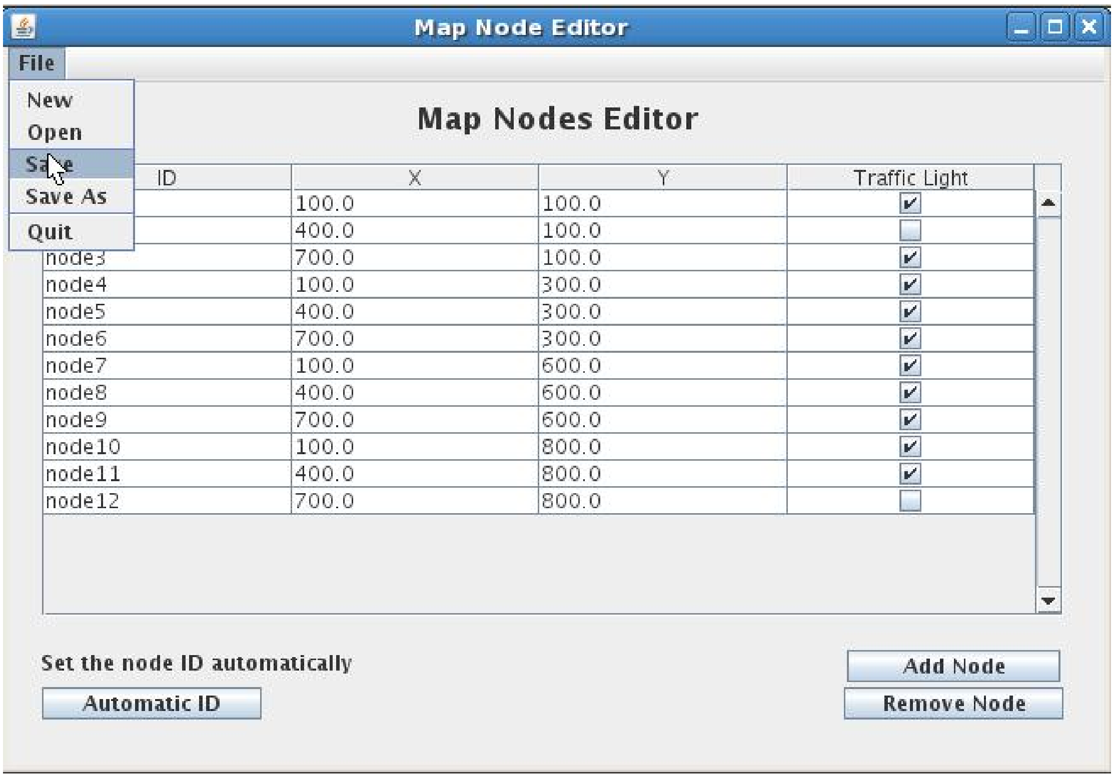


首先在 MOVE 的主選單上選擇”Node”，輸入下面的相關資訊，這些設定是根據圖一的範例所設計。



依照圖 1 設定的結果

等待一切輸入完成後，進行存檔的動作。



當你完成了編輯之後選擇‘File’ -> ‘save’ or ‘save as’，儲存的檔案名稱為<name>.nod.xml，在本實驗使用ex_NODE.nod.xml。其詳細的設定資訊如下：

<nodes>

<node id="node1" x="100.0" y="100.0" type="traffic_light"/>

<node id="node2" x="400.0" y="100.0" type="priority"/>

<node id="node3" x="700.0" y="100.0" type="traffic_light"/>

<node id="node4" x="100.0" y="300.0" type="traffic_light"/>

<node id="node5" x="400.0" y="300.0" type="traffic_light"/>

<node id="node6" x="700.0" y="300.0" type="traffic_light"/>

<node id="node7" x="100.0" y="600.0" type="traffic_light"/>

<node id="node8" x="400.0" y="600.0" type="traffic_light"/>

<node id="node9" x="700.0" y="600.0" type="traffic_light"/>

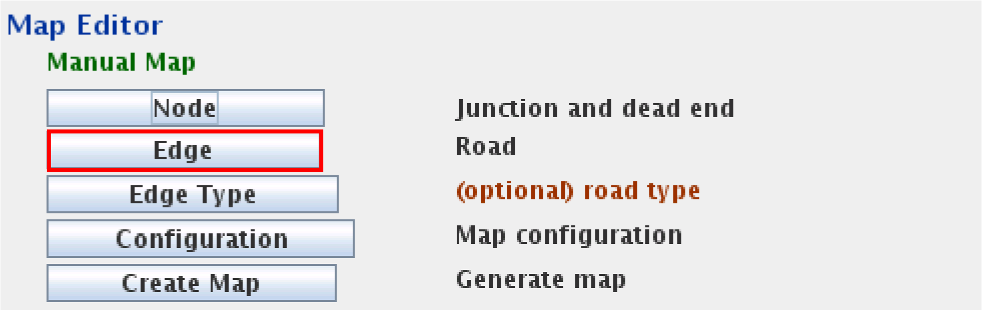
<node id="node10" x="100.0" y="800.0" type="traffic_light"/>

<node id="node11" x="400.0" y="800.0" type="traffic_light"/>

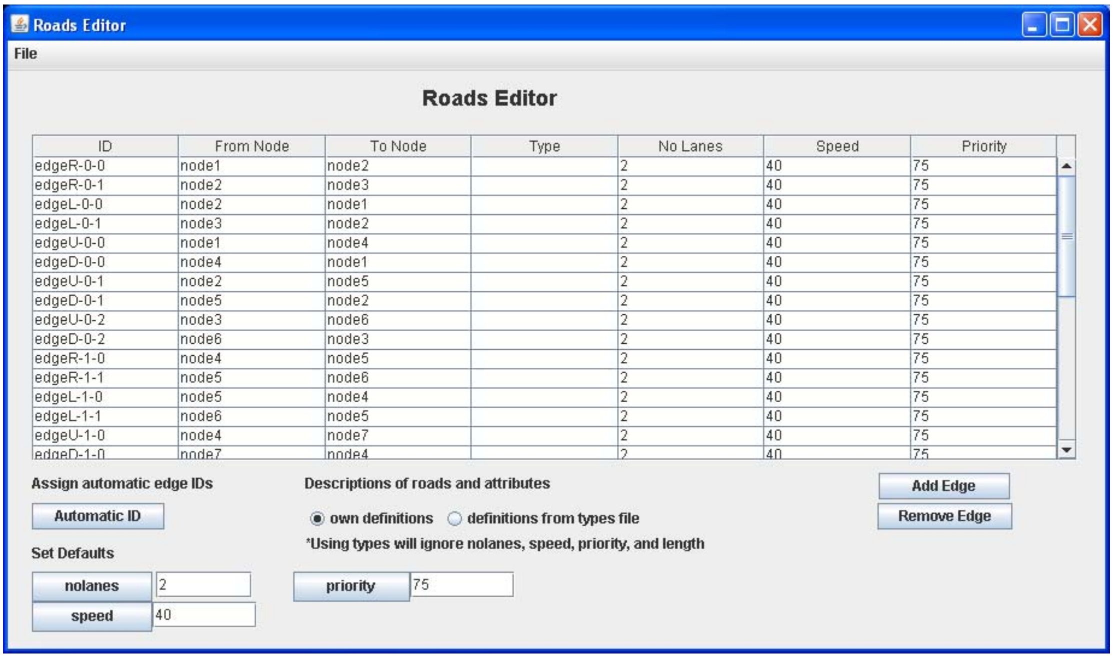
<node id="node12" x="700.0" y="800.0" type="priority"/>

</nodes>

步驟二. 手動設計地圖的街道 (ex_EDGE.edg.xml)



首先在MOVE的主選單上選擇”Edge”，輸入下面的相關資訊，在此你可以標記出所有的道路(一條道路將連接兩個之前創建的十字路口)。



ID	From Node	To Node	Type	No Lanes	Speed	Priority
edgeR-0-0	node1	node2		2	40	75
edgeR-0-1	node2	node3		2	40	75
edgeL-0-0	node2	node1		2	40	75
edgeL-0-1	node3	node2		2	40	75
edgeU-0-0	node1	node4		2	40	75
edgeD-0-0	node4	node1		2	40	75
edgeU-0-1	node2	node5		2	40	75
edgeD-0-1	node5	node2		2	40	75
edgeU-0-2	node3	node6		2	40	75
edgeD-0-2	node6	node3		2	40	75
edgeR-1-0	node4	node5		2	40	75
edgeR-1-1	node5	node6		2	40	75
edgeL-1-0	node5	node4		2	40	75
edgeL-1-1	node6	node5		2	40	75
edgeU-1-0	node4	node7		2	40	75
edgeD-1-0	node7	node4		2	40	75

依照圖 1 設定的結果。

完成後存檔<name>.edg.xml，在本實驗中我們使用檔名ex_EDGE.edg.xml。詳細</edges>的設定資訊如下：

<edges>

<edge id="edgeR-0-0" fromnode="node1" tonode="node2" priority="75" nolanes="2" speed="40" />

<edge id="edgeR-0-1" fromnode="node2" tonode="node3" priority="75" nolanes="2" speed="40" />

<edge id="edgeL-0-0" fromnode="node2" tonode="node1" priority="75" nolanes="2" speed="40" />

<edge id="edgeL-0-1" fromnode="node3" tonode="node2" priority="75" nolanes="2" speed="40" />

<edge id="edgeU-0-0" fromnode="node1" tonode="node4" priority="75" nolanes="2" speed="40" />
<edge id="edgeD-0-0" fromnode="node4" tonode="node1" priority="75" nolanes="2" speed="40" />
<edge id="edgeU-0-1" fromnode="node2" tonode="node5" priority="75" nolanes="2" speed="40" />
<edge id="edgeD-0-1" fromnode="node5" tonode="node2" priority="75" nolanes="2" speed="40" />
<edge id="edgeU-0-2" fromnode="node3" tonode="node6" priority="75" nolanes="2" speed="40" />
<edge id="edgeD-0-2" fromnode="node6" tonode="node3" priority="75" nolanes="2" speed="40" />

<edge id="edgeR-1-0" fromnode="node4" tonode="node5" priority="75" nolanes="2" speed="40" />
<edge id="edgeR-1-1" fromnode="node5" tonode="node6" priority="75" nolanes="2" speed="40" />
<edge id="edgeL-1-0" fromnode="node5" tonode="node4" priority="75" nolanes="2" speed="40" />
<edge id="edgeL-1-1" fromnode="node6" tonode="node5" priority="75" nolanes="2" speed="40" />

<edge id="edgeU-1-0" fromnode="node4" tonode="node7" priority="75" nolanes="2" speed="40" />
<edge id="edgeD-1-0" fromnode="node7" tonode="node4" priority="75" nolanes="2" speed="40" />
<edge id="edgeU-1-1" fromnode="node5" tonode="node8" priority="75" nolanes="2" speed="40" />
<edge id="edgeD-1-1" fromnode="node8" tonode="node5" priority="75" nolanes="2" speed="40" />
<edge id="edgeU-1-2" fromnode="node6" tonode="node9" priority="75" nolanes="2" speed="40" />
<edge id="edgeD-1-2" fromnode="node9" tonode="node6" priority="75" nolanes="2" speed="40" />

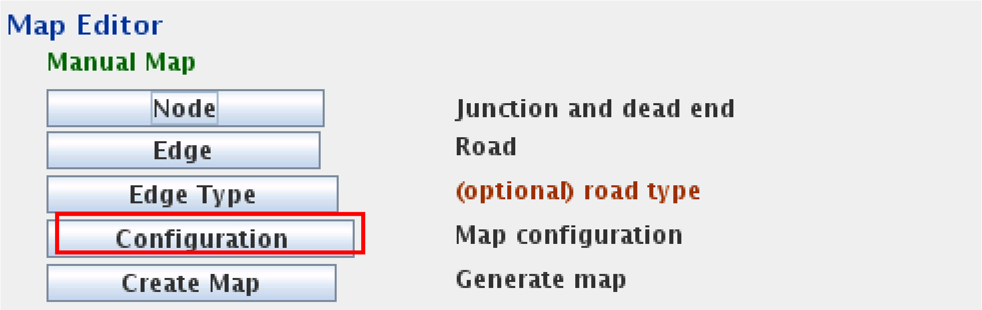
<edge id="edgeR-2-0" fromnode="node7" tonode="node8" priority="75" nolanes="2" speed="40" />
<edge id="edgeR-2-1" fromnode="node8" tonode="node7" priority="75"

nolanes="2" speed="40" />
<edge id="edgeL-2-0" fromnode="node8" tonode="node9" priority="75" nolanes="2" speed="40" />
<edge id="edgeL-2-1" fromnode="node9" tonode="node8" priority="75" nolanes="2" speed="40" />

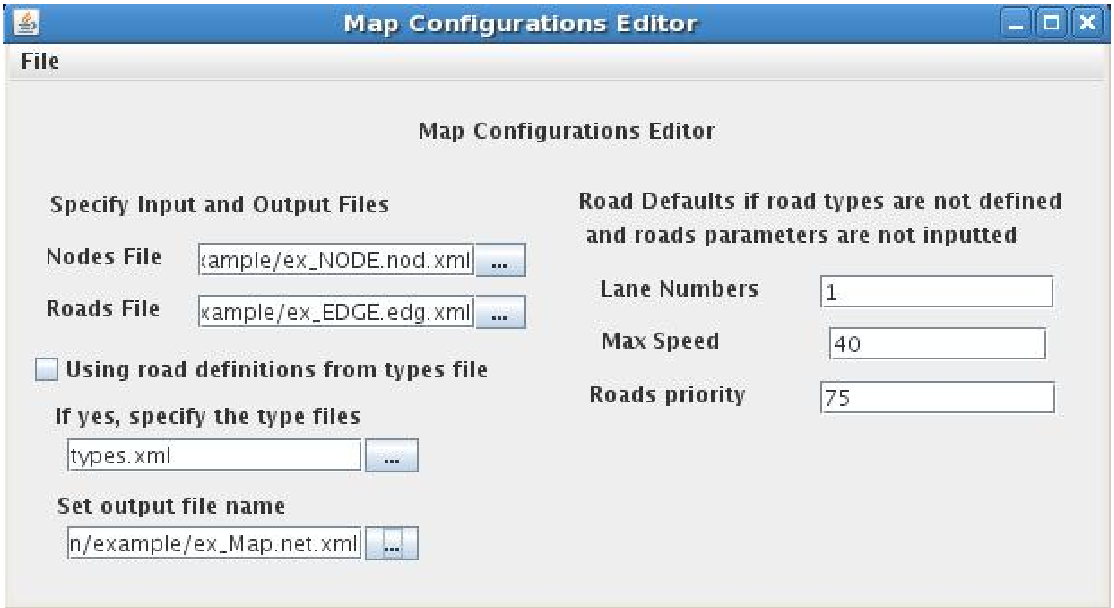
<edge id="edgeU-2-0" fromnode="node7" tonode="node10" priority="75" nolanes="2" speed="40" />
<edge id="edgeD-2-0" fromnode="node10" tonode="node7" priority="75" nolanes="2" speed="40" />
<edge id="edgeU-2-1" fromnode="node8" tonode="node11" priority="75" nolanes="2" speed="40" />
<edge id="edgeD-2-1" fromnode="node11" tonode="node8" priority="75" nolanes="2" speed="40" />
<edge id="edgeU-2-2" fromnode="node9" tonode="node12" priority="75" nolanes="2" speed="40" />
<edge id="edgeD-2-2" fromnode="node12" tonode="node9" priority="75" nolanes="2" speed="40" />

<edge id="edgeR-3-0" fromnode="node10" tonode="node11" priority="75" nolanes="2" speed="40" />
<edge id="edgeR-3-1" fromnode="node11" tonode="node10" priority="75" nolanes="2" speed="40" />
<edge id="edgeL-3-0" fromnode="node11" tonode="node12" priority="75" nolanes="2" speed="40" />
<edge id="edgeL-3-1" fromnode="node12" tonode="node11" priority="75" nolanes="2" speed="40" />
</edges>

步驟三. 地圖檔組態設定 (ex_Map.netc.cfg)



當我們完成了”Node”與”Edge”兩個部份的建置之後，在主選單上選擇”Configuration”，進入地圖的組態設定介面。



在這邊需載入前面步驟一(Node)與步驟二(Edge)的設定檔，如果使用者沒有在”Edge”部分進行詳細的設定，這邊可以使用預設的選項進行整體設定，如果考量的道路等級都相同的情況下，這樣的設計會快速許多。這邊我們設定的輸出檔為”ex_Map.net.xml”，然後儲存檔案為<name>.netc.cfg，在本實驗中我們使用 ”ex_Map.netc.cfg”。而設定檔詳細的資訊如下：

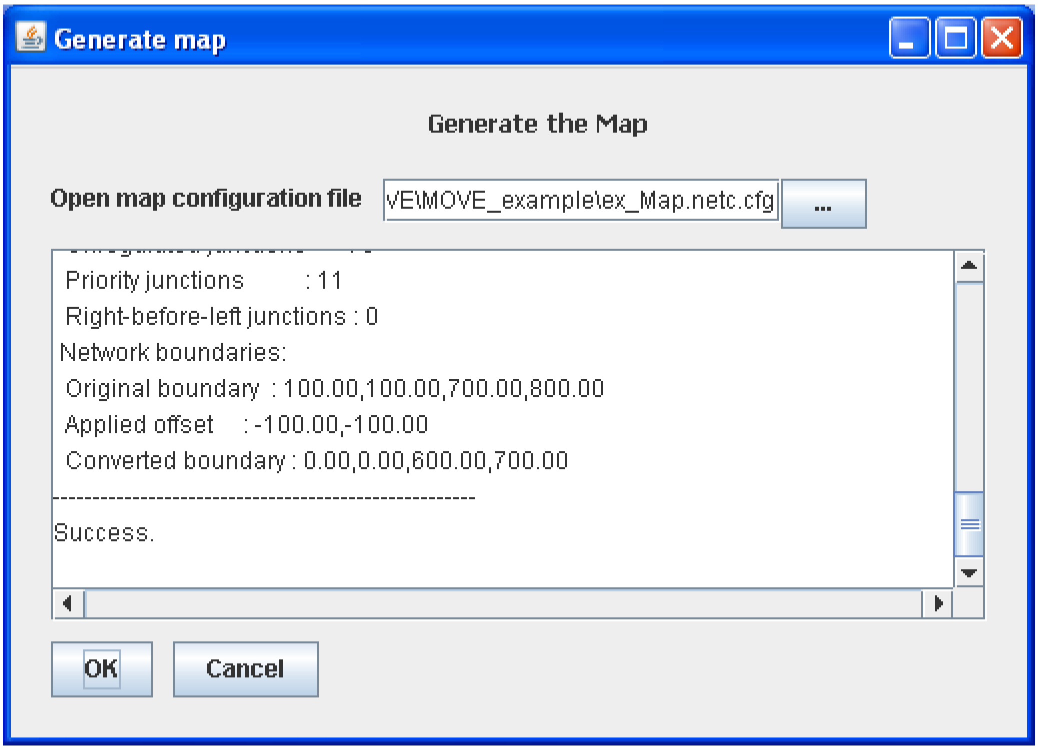
<configuration>

<input
xml-node-files="C:\For_MOVE\MOVE_example\ex_NODE.nod.xml"
xml-edge-files="C:\For_MOVE\MOVE_example\ex_EDGE.edg.xml"
xml-connection-files=""
type-file=""

```
    />

    <output
        output-file="C:\For_MOVE\MOVE_example\ex_Map.net.xml"
    />

    <defaults
        type="Unknown"
        lanenumber="1"
        speed="40"
        priority="75"
        capacity-norm=""
    />

    <reports
        print-options="false"
    />

    <processing
        speed-in-kmh="false"
        no-turnarounds="false"
        remove-geometry="x"
    />

</configuration>
```

步驟四. 產生地圖 (ex_Map.net.xml)

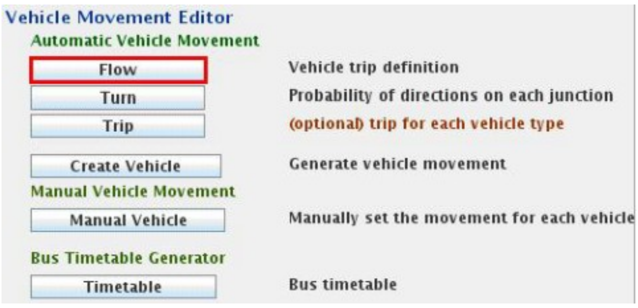
當我們完成了組態檔的建置後，我們在主選單上面選擇”Create Map”。



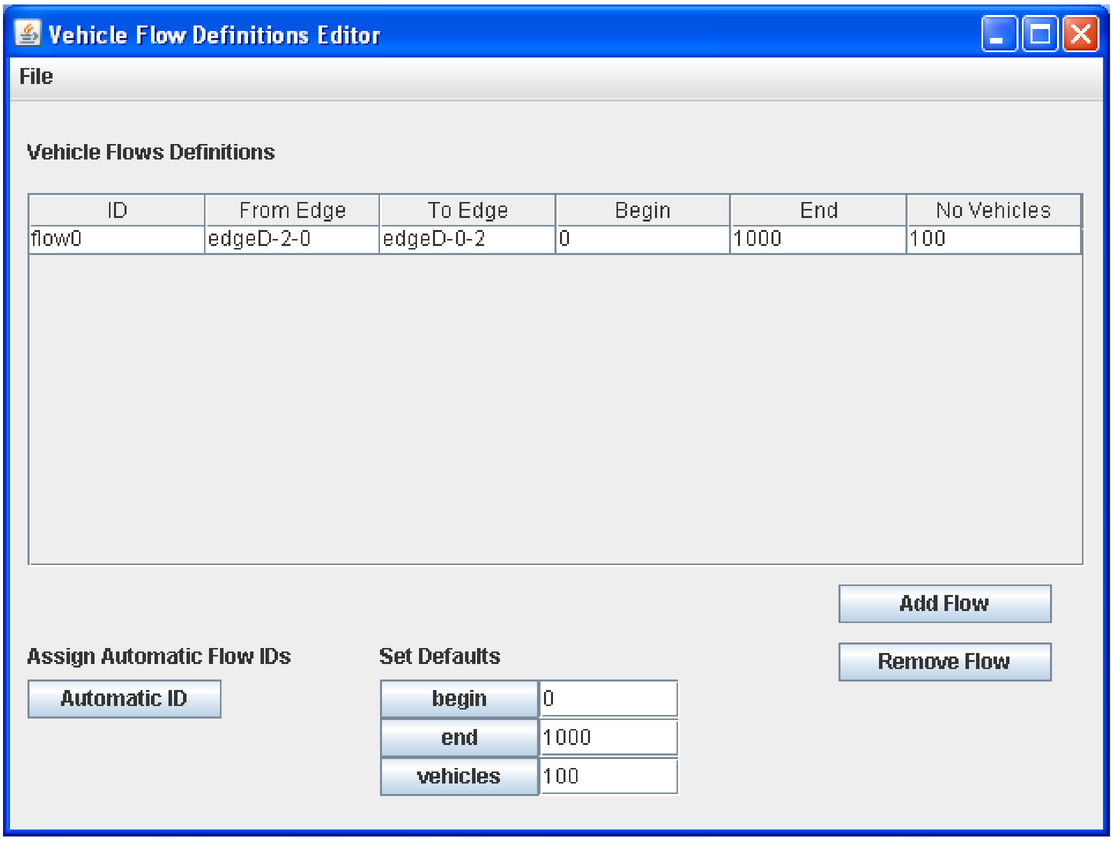
這邊只需要載入我們剛儲存好的地圖設定檔”ex_Map.netc.cfg”，然後按 OK，而在設定檔上面設定好的地圖輸出檔即會自動產生，此檔就是地圖檔。

II. 車輛移動產生

步驟一. 使用車流(Flow)編輯器 (ex_FLOW.flow.xml)



在主選單上面選擇"Flow"，會開啟車流編輯器，使用者可以在這邊設定車流移動的路徑。



依照圖 1 設定的結果

在這部份為群組車流的設定，車輛的移動將會隨著設定的兩個路段作為起始點與終點。例如：

edge1 (連接至 node 1 與 2)

edge2 (連接至 node 2 與 3)

edge3 ((連接至 node 3 與 4)

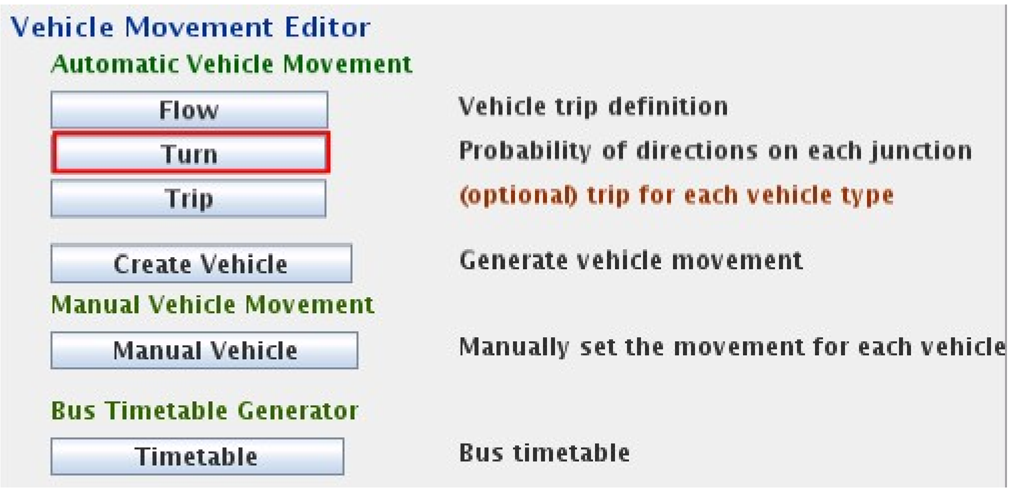
如果要讓車子移動從 node1 移動到 node2 在移動到 node3，你必須設定起始點為

edge1，終點為 edge3，而並非是 edge1 至 edge2。車流建置完成後存檔，儲存的檔案為<name>.flow.xml，在本實驗中我們使用 “ex_FLOW.flow.xml”。而詳細的設定資訊如下：

<flows>
<flow id="flow0" from="edgeD-2-0" to="edgeD-0-2" begin="0" end="1000" no="100" />
</flows>

另外對於 Turn 與 Trip 在本範例中不進行描述，有興趣的使用者亦可自行摸索，其中 Turn 必須設置在每個節點的轉彎機率，這邊是具方向性的。

步驟二. 設定接點轉彎機率(此範例不使用)

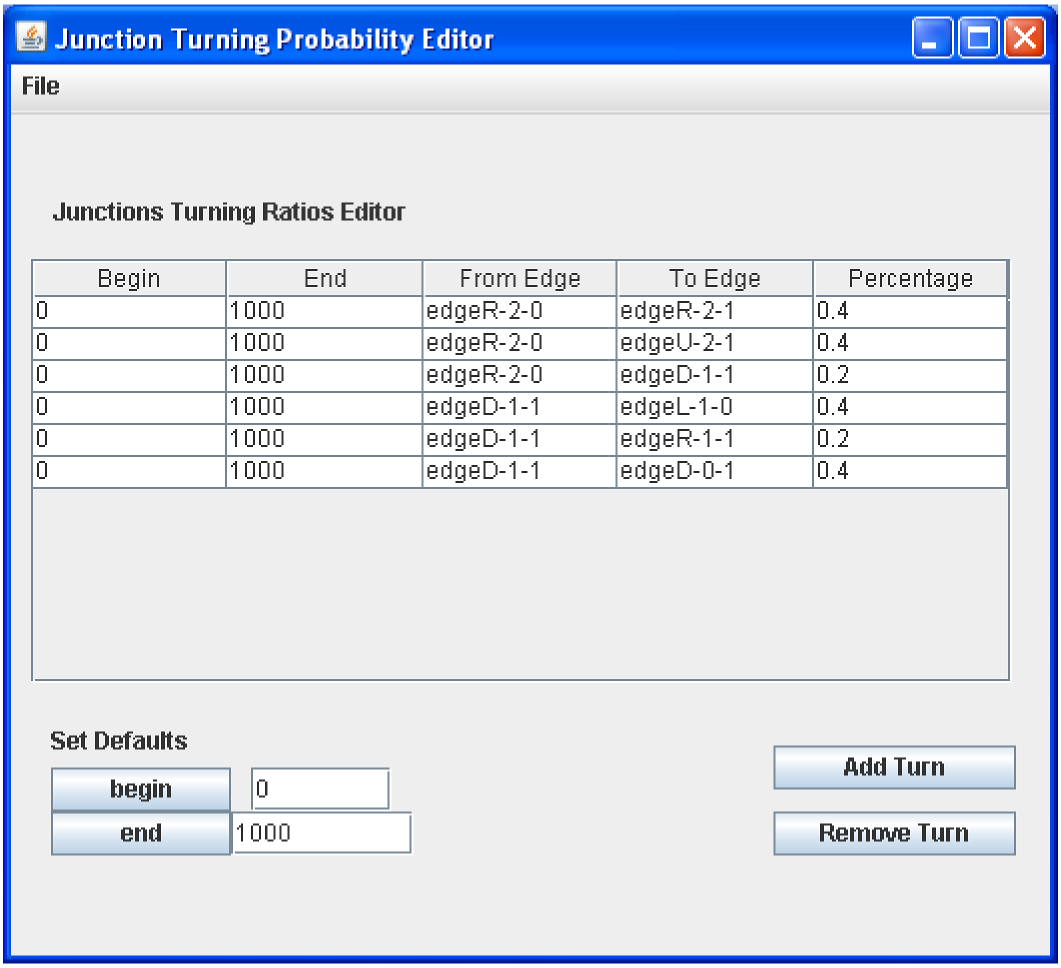


在 MOVE 主選單上選者”Turn”，將可以設定在每個十字路口車輛轉彎的機率，你必須在之前車輛移動中就設定好定義，並且在此編輯器中，所有方向轉彎機率的合必須為 100%。

備註:若想在 MOVE 設定相同的機率，可以嘗試使用以下指令:

sumo-jtrrouter -n ex_Map.net.xml -f ex_FLOW.flow.xml -t ex_TURN.turn.xml -v -o ex_ROU.rou.xml -b 0 -e 1000 --continue-on-unbuild --turn-defaults=30,30,40

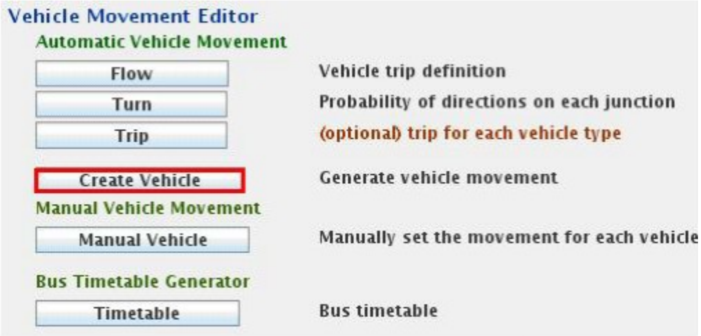
此指令會產生特定的機率格式:左轉，直走，右轉。



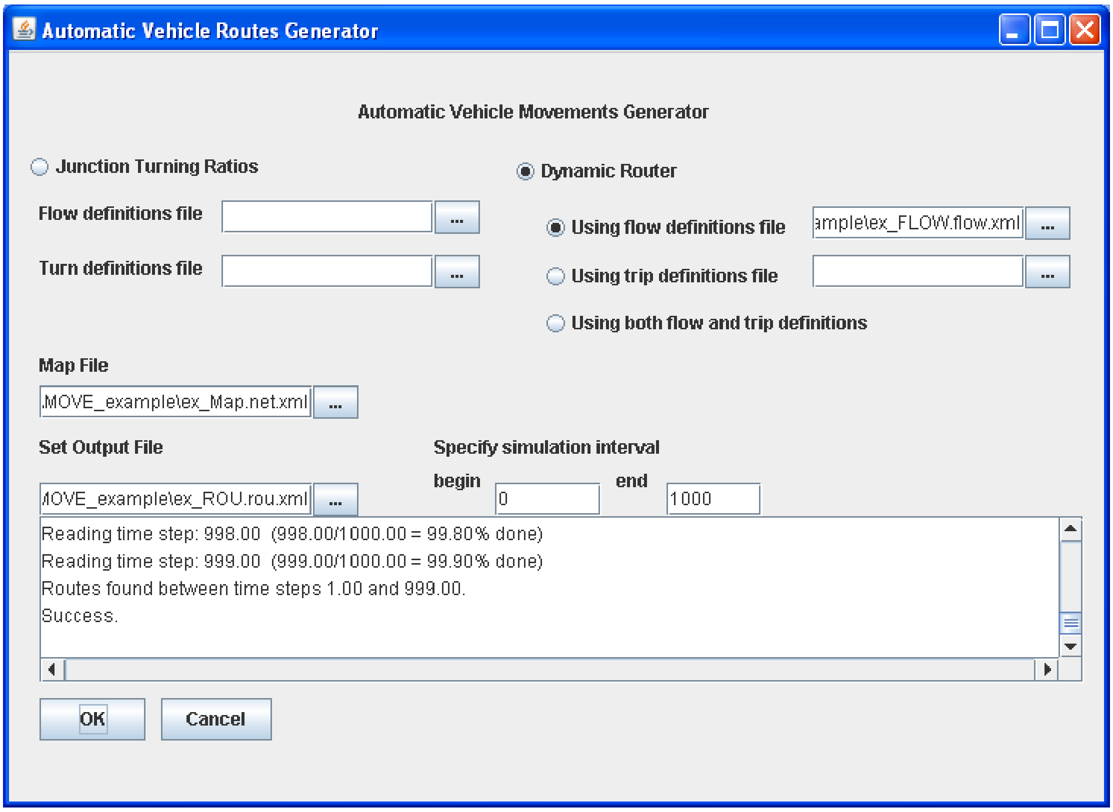
ex_TURN.turn.xml 的設定內容如下:

<turn-defs>
<interval begin="0" end="1000">
<fromedge id="edgeR-2-0">
<toedge id="edgeR-2-1" probability="0.4"></toedge>
<toedge id="edgeU-2-1" probability="0.4"></toedge>
<toedge id="edgeD-1-1" probability="0.2"></toedge>
</fromedge>
<fromedge id="edgeD-1-1">
<toedge id="edgeL-1-0" probability="0.4"></toedge>
<toedge id="edgeR-1-1" probability="0.2"></toedge>
<toedge id="edgeD-0-1" probability="0.4"></toedge>
</fromedge>
</interval>
</turn-defs>

步驟三. 產生車輛移動路徑 (ex_ROU.rou.xml)



當完成了車流編輯之後，在主選單上面選擇”Create Vehicle”，這邊可以產生車輛的移動路徑檔。

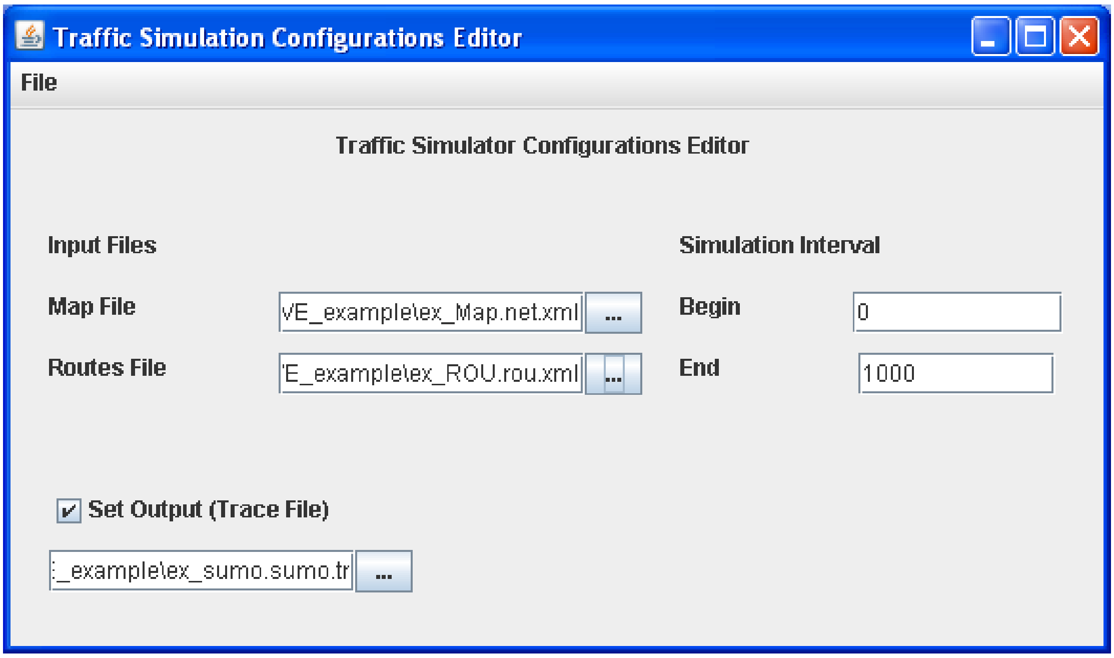


首先選擇我們在建置好的地圖檔”ex_Map.net.xml”，並套用我們前面產生的車流檔 “ex_FLOW.flow.xml”，再指定移動路徑的輸出檔案位置與名稱 <name>.rou.xml，本實驗中我們以”ex_ROU.rou.xml”為例子。設定好之後按 OK，即會自動產生移動的路徑。

當我們完成了車輛移動路徑的設定檔後，相信使用者已經了解到 Move module 的初步運作方式，然而透過自動產生的 Flow 幾乎都是相同的，所以 MOVE 也提供了手動輸入的方式來讓使用者自行指定車輛的行走方式。接下來我們要使用已經產生好的路網地圖與車流資訊來產生一個實際的 mobility patterns。

步驟四. 車輛移動模擬建立 (ex_SUMOTRACE.trace and ex_SUMO.sumo.cfg)

當路網地圖與車流的移動均完成後，我們需要建置車輛移動模擬的組態檔，在主選單上面選擇”Configuration“進入設定的介面。其中”Begin”與”End”為車輛模擬的時間軸。



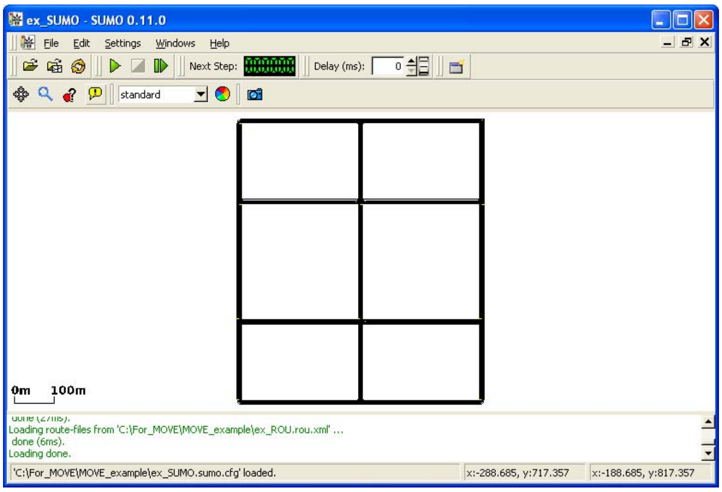
這部份需要載入前面設置好的地圖檔“ex_Map.net.xml”與車輛移動路徑的設定檔”ex_ROU.rou.xml”，並指定模擬時間長短，這邊由於我們需要使用 NS-2 進行網路效能模擬，所以開啟 Trace File 並指定檔案名稱<name>.move.trace，在本實驗中我們使用檔案名稱”ex_SUMOTRACE.trace”紀錄 mobility patterns，最後選取 ”File” 然後儲存設定檔<name>.sumo.cfg，在本實驗我們儲存檔名為”ex_SUMO.sumo.cfg”。其詳細的組態檔資訊如下：

```
<configuration>

    <input
       net-file="C:\For_MOVE\MOVE_example\ex_Map.net.xml"
       route-files="C:\For_MOVE\MOVE_example\ex_ROU.rou.xml"
       additional-files=""
       junction-files=""
    />
```

```
    <output
      netstate-dump="C:\For_MOVE\MOVE_example\ex_sumo.sumo.tr"
      tripinfo-output="output-tripinfos.xml"
      emissions-output="output-emissions.xml"
      vehroute-output="output-vehroutes.xml"
    />

    <time
      begin="0"
      end="1000"
      time-to-teleport="-1"
      srand="23423"
      route-steps="-1"
    />

    <reports
        print-options="false"
    />

</configuration>
```

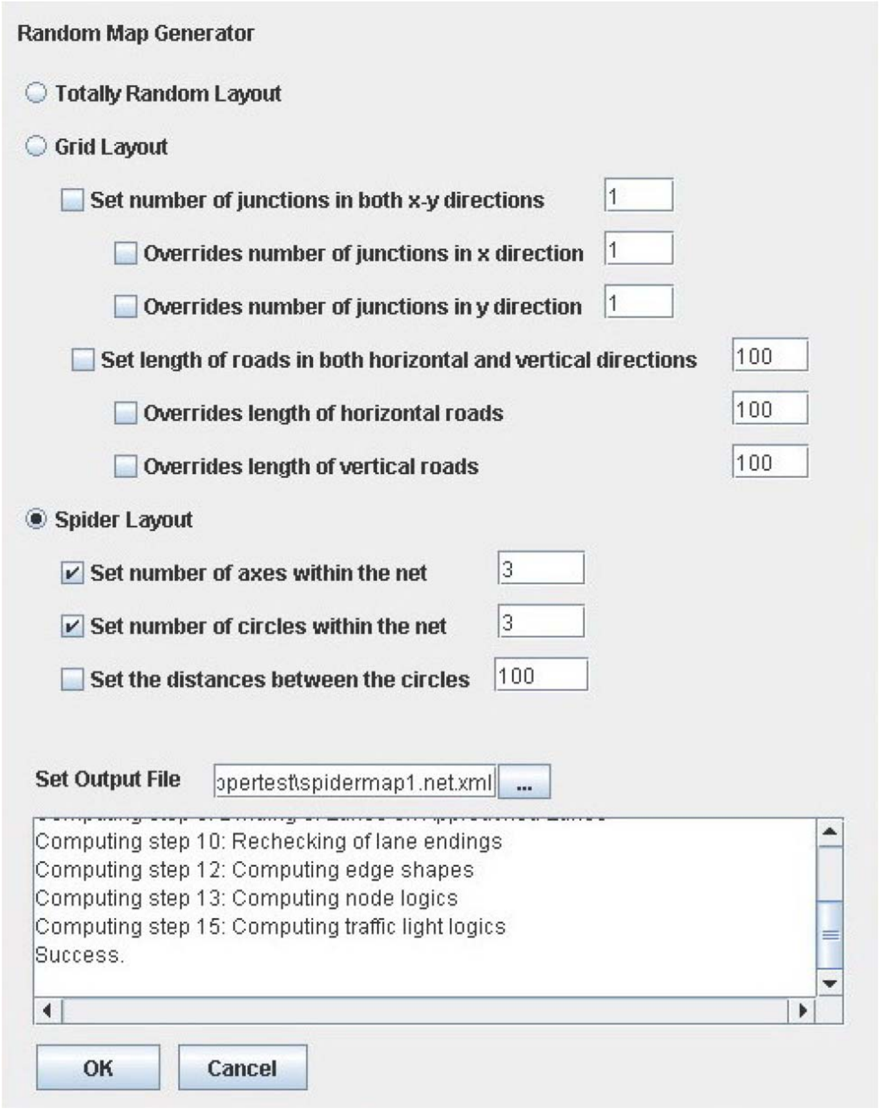
步驟五. 圖形化介面模擬

當你完成了上述的每一個步驟之後，現在你可以選擇“Visualization”來觀察車輛移動的情形。

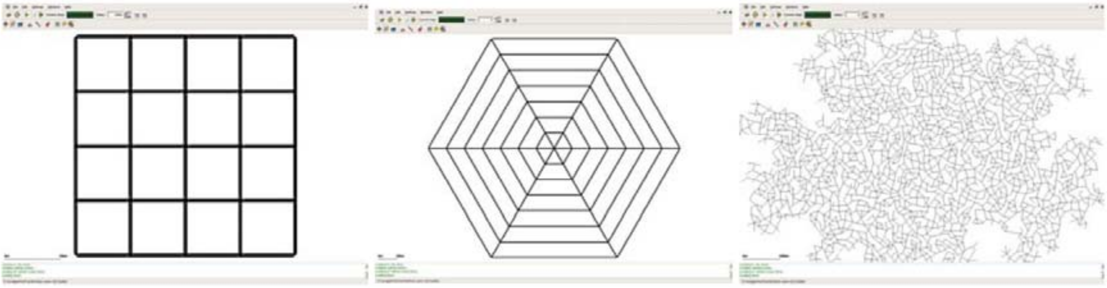


關於 SUMO 的詳細使用與定義，使用者可以參閱 SUMO 的使用手冊。

III. 自動產生地圖 (本範例不使用)



在 MOVE 主選單中選擇”Random Map”來啟動此功能。你可以客製化的自動創建三種類型的佈局:網格、蜘蛛網、隨機。



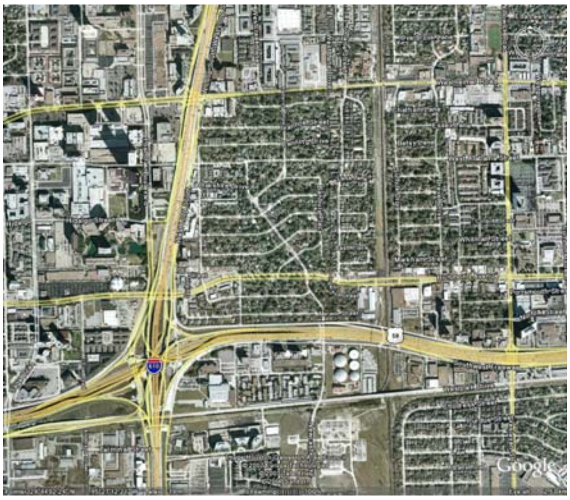
選定你想要的佈局後，設定目標檔名<name>.net.xml，接著選 OK，檔名為<name>.net.xml 的地圖檔將會自動被創建。

IV. 以 Tiger Line Map 檔產生地圖

若持有 Tiger Line Map 檔，你將可以把它轉換為 MOVE map。在 MOVE 主選單上選擇”Convert Tiger”，選擇指定的 Tiger file (通常為<name>.dat)，在指定儲存檔名。

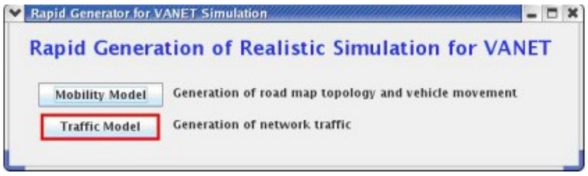
警告：請勿加上任何副檔名 i.e. /home/tigeroutput (無副檔名).將會產生<name>.nod.xml, <name>.edg.xml, <name>.netc.cfg, and <name>.net.xml (your map file)

備註:當下版本的 MOVE 將把在 TIGER LINE file 中所有種類的路段是為相同類型的路段(可以打開<name>.edg.xml 檔確認)，若要支援更多類型的路段，請修改原始檔 tiger.java。

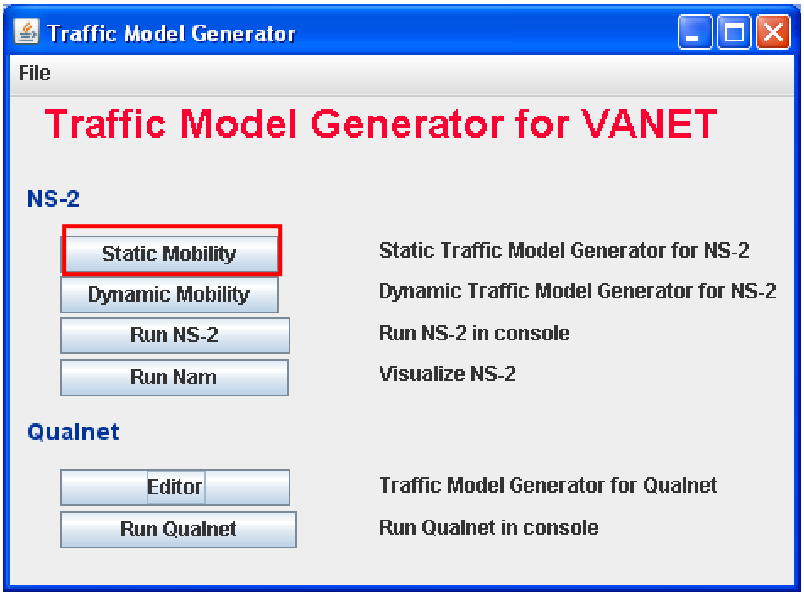


google map 與 TIGER line map

網路模組產生器(Traffic generation)



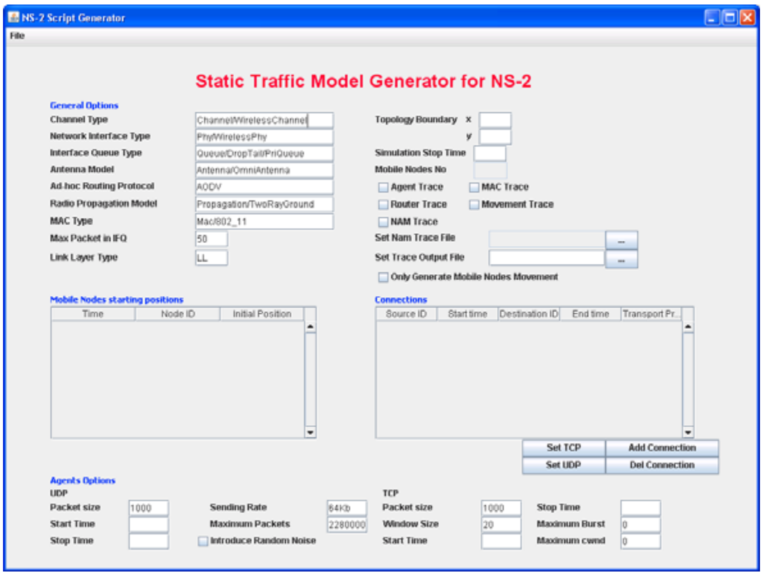
首先在 MOVE 的主選單上面選擇”Traffic Model”進入 traffic model 的設定介面。



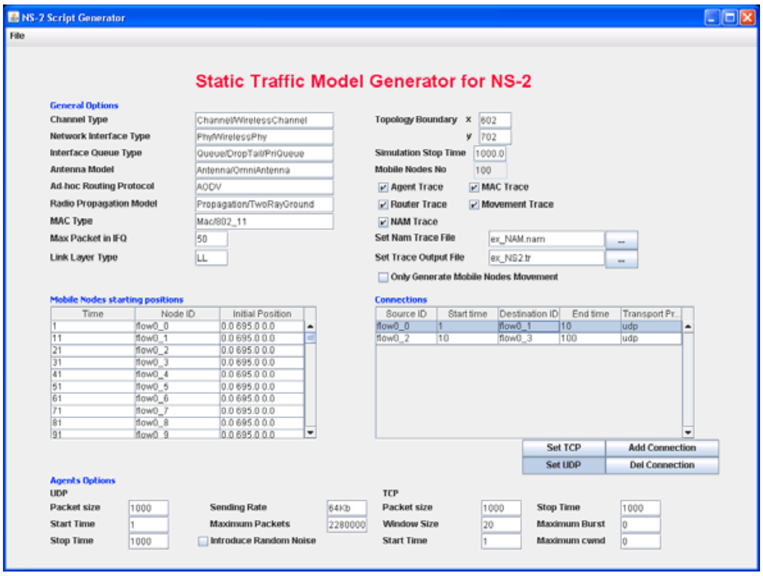
MOVE 在 traffic model 產生器有兩大部分：分別為 NS-2 與 Qualnet，在本實驗中我們使用 NS-2 為範例。至於 NS-2 的詳細使用方式，可以參照 NS-2 的使用手冊。此範例指使用靜態移動模型。

NS-2 腳本編輯器 (靜態移動模型)

步驟一. 產生 NS-2 的劇本檔 (ex_NS-2.tcl)



透過 NS-2 的 Traffic model Generator 編輯器可以快速的產生 NS-2 的模擬劇本設定檔(即 TCL 檔)，首先載入 SUMO 的 trace 檔 ”ex_SUMOTRACE.trace”與地圖檔”ex_Map.net.xml”，提供產生器所需的資訊。



等待產生器讀取資料(這個部份的執行時間會依據檔案大小以及電腦的硬體效能而有所不同)，當讀取完之後，我們可以為 TCL 模擬器設定一些額外的細項設定，關於 TCL 的詳細設定使用者可以參閱 NS-2 的使用手冊。 此實驗我們使用 NAM trace，並指定存檔名稱，並指定檔案位置，接下來我們可在左方表格加入車輛連結關係以及分配通訊協定。

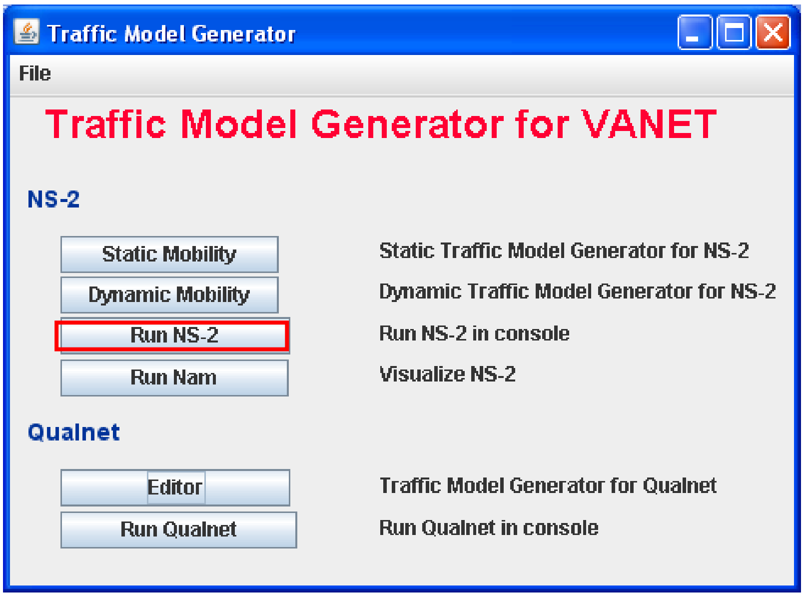
在“mobile nodes starting positions”的表格中，你可以看到不同的時間、ID、起始位置，你可以參考此資訊在右方增加網路連線的設定。

注意，節點 ID 與 SUMO 中車流量的名字有關，舉例來說，若此車流量名為 test0，那節點的 ID 將會命名為 test0_0、test0_1、test0_2……、test0_x(x 為此車流的編號)。請確定節點 ID 無誤，如果你輸入了一個不存在的 ID，TCL 檔將無法保存正確的 ID。

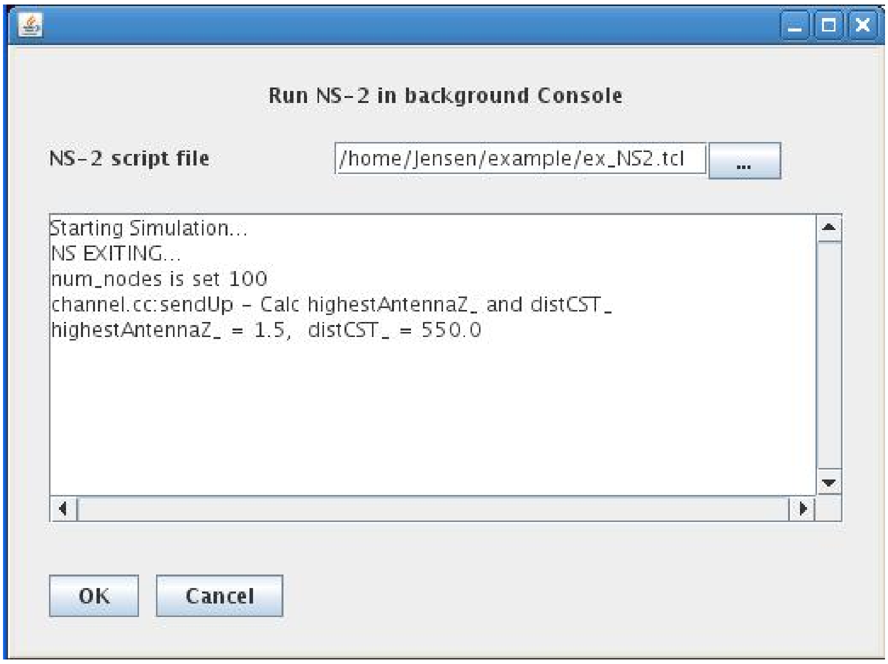
當以上工作完成後，選擇 File->Save 或 Save As，然後輸入檔名<name>.tcl (ex_NS2.tcl)

備註:請在 Linux 平台上使用 ns-2。

步驟二. 執行 NS-2 於背景模式 (background console)(ex_NAM.nam and ex_NS-2.tr)

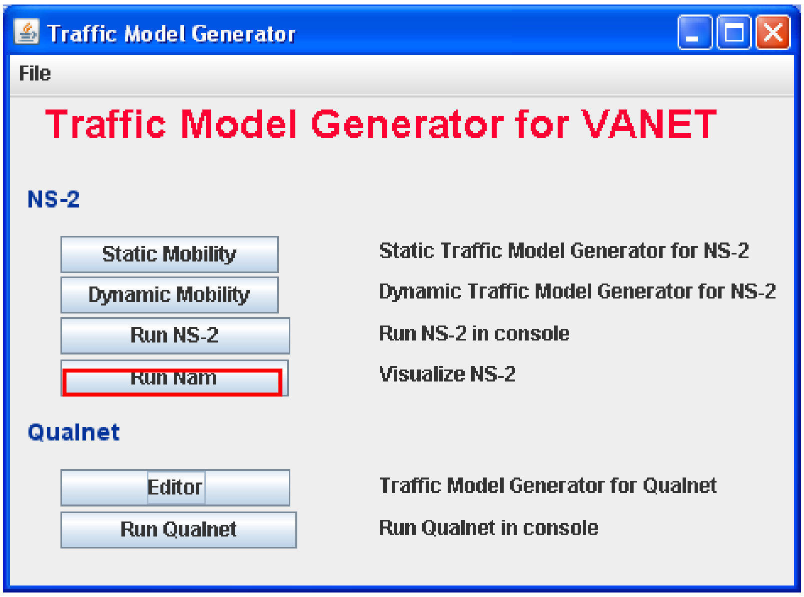


當使用者完成了 NS-2 劇本檔的建置之後，執行”Run NS-2”。指定前面範例建置好的 NS-2 劇本檔” ex_NS-2.tcl”，然後按 ok 執行。

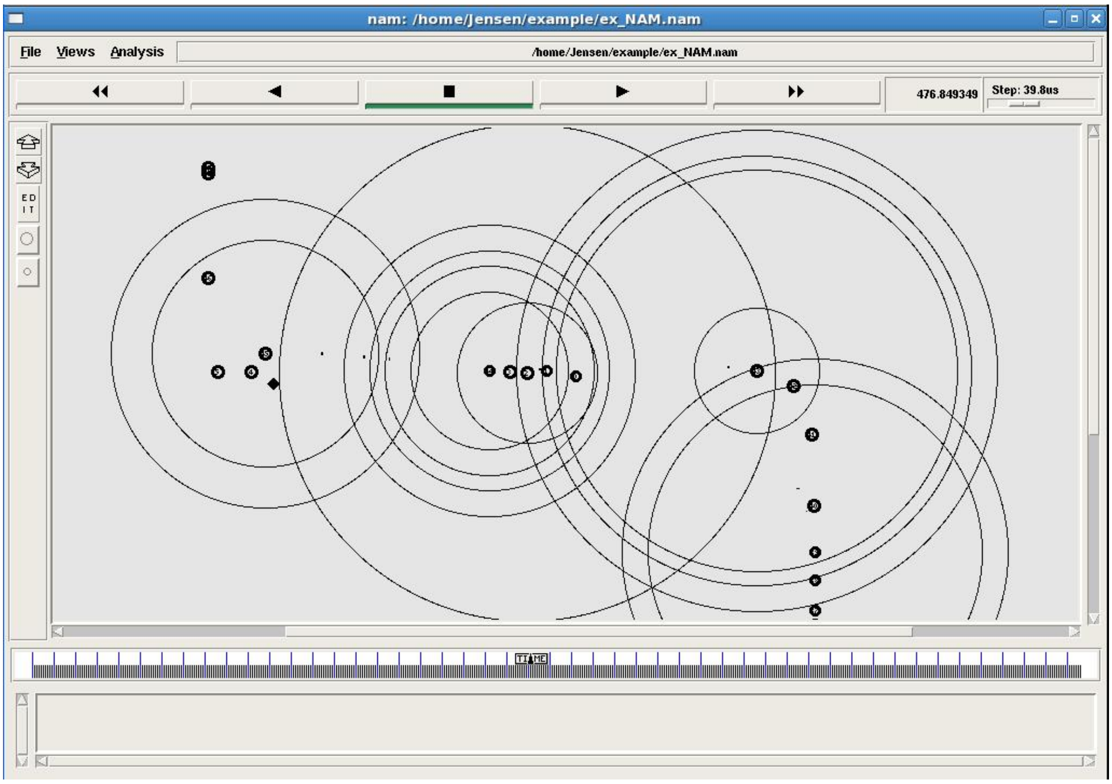


你將可以在你自己的環境中或是此軟體的 NS-2 script runner 中(在 traffic model main menu 中選擇"Run NS-2")執行此 NS-2 腳本。

步驟三. 執行 NAM visualize NS-2(只能在 Linux 平台執行)

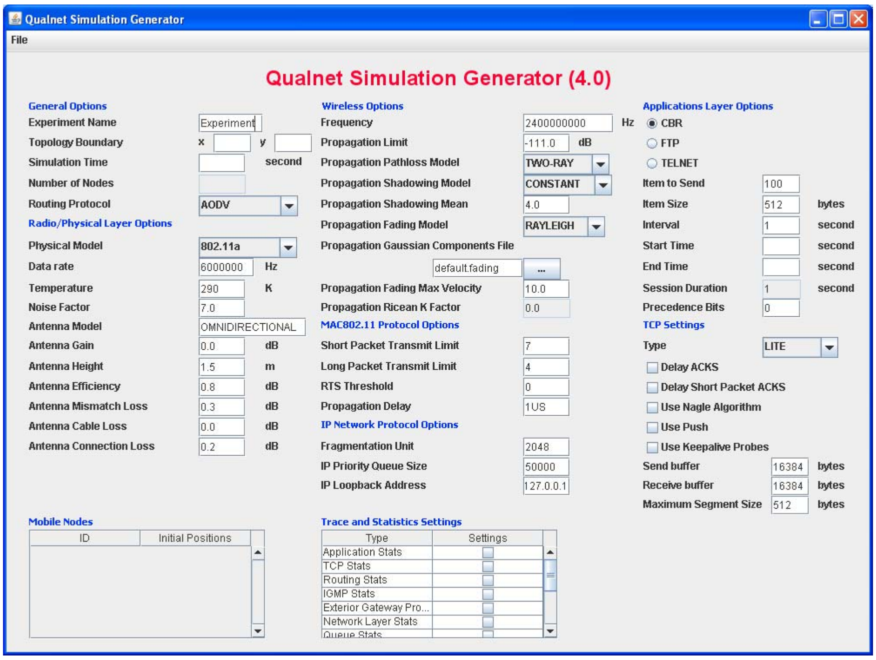


最後，在選單中選擇"Run NS-2" runs NS-2 in console。

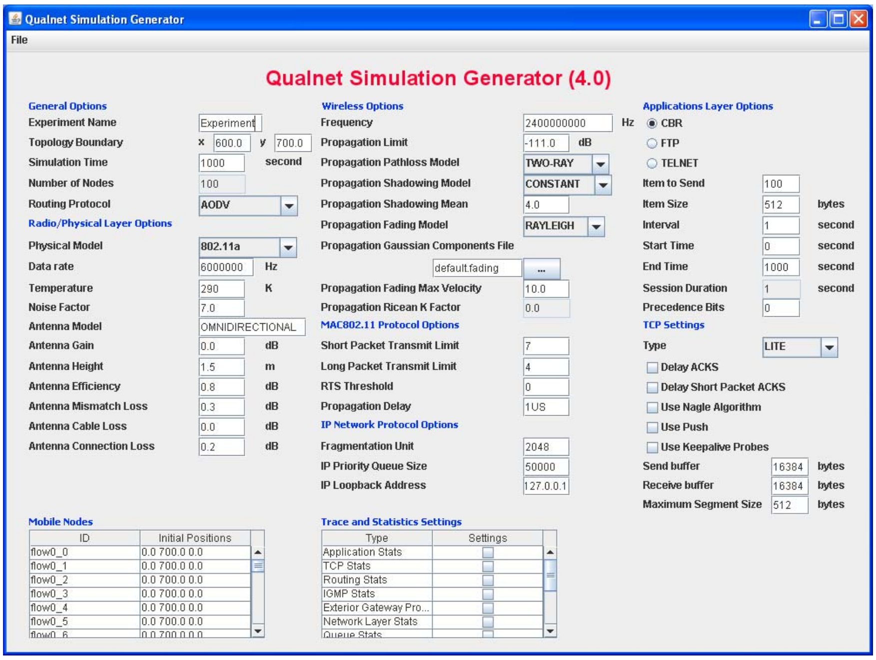


現在我們可以播放 NAM 觀察範例實際動作的情形。

Qualnet 腳本編輯器



此編輯器將產生 Qualnet 模擬器工具的模擬腳本(設定檔)。首先，匯入 MOVE Trace 檔 (i.e.ex_SUMOTRACE.sumo.tr)和地圖檔.net.xml (i.e. ex_Map.net.xml)。



等待檔案讀取(這將與輸入檔案大小與電腦硬體配備有關)，當讀取結束，你可以設定模擬器選項，若需要更多細節說明，請參考 Qualnet 說明手冊。在此我們只產生 mobility pattern in Qualnet，我們也計畫在未來能創造網路連結編輯器。

Appendix I: 紅綠燈號誌控制

參考網址:

http://sourceforge.net/apps/mediawiki/sumo/index.php?title=Advanced_Traffic_Lights_Simulation

首先，將產生好的地圖檔以文字編輯器開啟，將紅綠燈的定義部分複製起來(搜尋">YOUR_TLS_NAME<")，且複製在<tl-logic>標籤內的部分，將他複製到另一檔案並以<additional>標籤包住。

你的儲存內容應如下(本範例檔名為“**traffic_duration.add.xml”)**：

<additional>

CUT_TLS_DEFINITION　//欲更該的紅綠燈資訊

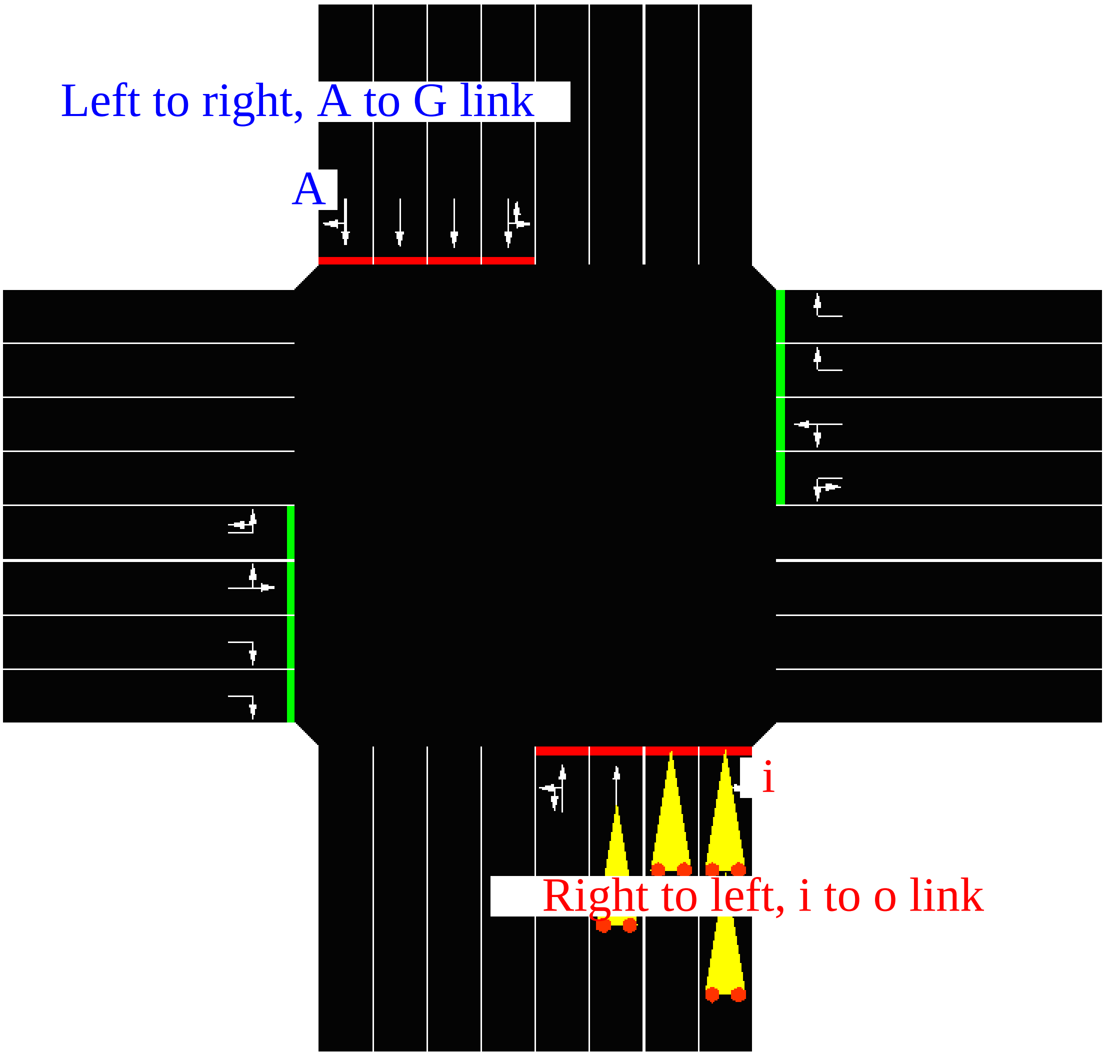
</additional>

完成以後，將”programID”以其他文字取代，你可以改變所有時相的間隔，你可以讀取此文件只要將他的檔名改為對應的副檔名。

舉例來說，若要改變node8的週期，你可以在map檔中獲得以下資訊：

檔名: *traffic_duration.add.xml*

```
<additional>
    <tl-logic id="node8" type="static" programID="modify" offset="0">
        <phase duration="35" state="GGGGGggrrrrrrrGGGGGggrrrrrrr"/> //Green light Index A to index G highlight with blue color as shown in under Figure. Index I to index O highlight with red color.
        <phase duration="5" state="yyyyyggrrrrrrryyyyyggrrrrrrr"/> //Yellow light
        <phase duration="6" state="rrrrrGGrrrrrrrrrrrrGGrrrrrrr"/> //Green light for turn left
        <phase duration="5" state="rrrrryyrrrrrrrrrrrryyrrrrrrr"/> // Yellow light for turn left
        <phase duration="31" state="rrrrrrrGGGGgggrrrrrrrGGGGggg"/> // Similar as above configuration, just different directions.
        <phase duration="5" state="rrrrrrryyyygggrrrrrrryyyyggg"/>
        <phase duration="6" state="rrrrrrrrrrrGGGrrrrrrrrrrrGGG"/>
        <phase duration="5" state="rrrrrrrrrrryyyrrrrrrrrrrryyy"/>
    </tl-logic>
```



TAIWAN, NCKU university, CSIE, LENS lab.
Updater: Chine-Ming Chou

檔名: *ex_SUMO.sumo.cfg*

```
<configuration>

    <input>
        <net-file value="/home/Jensen/example/ex_Map.net.xml"/>
        <route-files value="/home/Jensen/example/ex_Map.net.xml"/>
        <additional-files value="/home/Jensen/example/traffic_duration.add.xml "/>
        <junction-files value=""/>
    </input>

    <output>
        <netstate-dump value="/home/Jensen/example/grid.sumo.tr"/>
        <tripinfo-output value="output-tripinfos.xml"/>
        <emissions-output value="output-emissions.xml"/>
        <vehroute-output value="output-vehroutes.xml"/>
    </output>

    <time>
        <begin value="0"/>
        <end value="1000"/>
        <time-to-teleport value="-1"/>
        <srand value="23423"/>
        <route-steps value="-1"/>
    </time>

    <reports>
          <print-options value="false"/>
    </reports>

</configuration>
```

備註: 此部分感謝SUMO的開發人員 Daniel 的指導。

Appendix II: 隨機路線

由於 SUMO 移除了隨機路線產生功能，我們在此介紹 script for generating random trips 來取而代之，此方法須使用 Python。

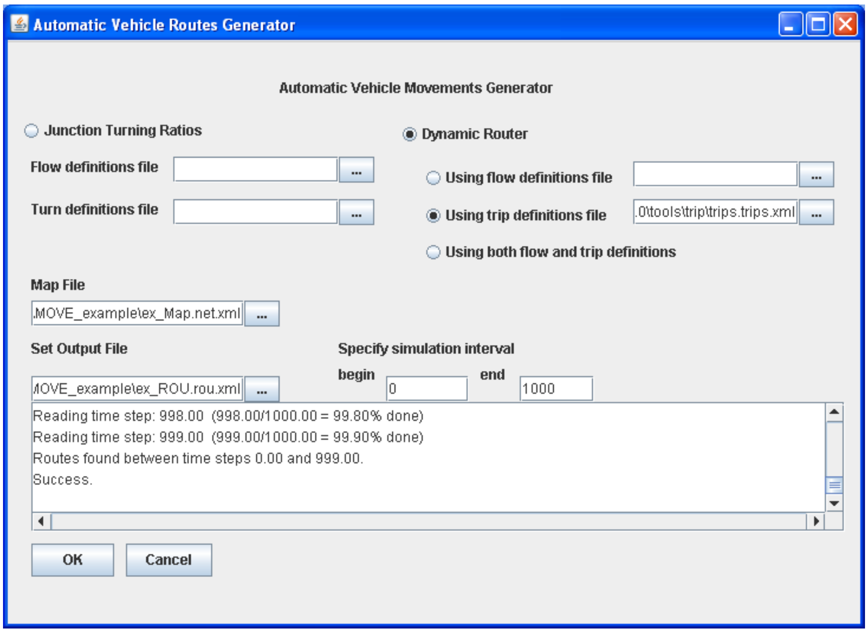
- Python 2.7 or later - http://www.python.org/download/

此腳本位於: <***SUMO***>/tools/trip

"**randomTrips.py**"可產生一組隨機路徑對應所指定的網路(指令“-n”)，可以隨機選擇起點與終點或是以長度考量(指令”-l”)，或是以車道數目(指令”-L”)，亦可兩者皆使用。結果的路線將儲存在指定的 xml 檔(指令”-o”，預設檔名為”trips.trips.xml”)與 DUAROUTER(指令”-r”加上結果的 route 檔)相容。路線將會平均的分布在一區間，此區間的開頭以指令”-b”(預設 0)而結束時間以指令”-e”(預設 3600)以秒為單位表示。路線的數量將以重複頻率定義(指令”-p” 預設為 1)以秒為單位。每個路線都有一個 id 以一字首開頭(指令”-t” 預設為”t”)接著依序的號碼，範例如下：

```
randomTrips.py -n input_net.net.xml -e 1000 –l
```

備註：此腳本不會檢查此目的地是可以到達的，這項工作將由路由器處理。當此 trips 檔產生後，你可以使用 MOVE 產生相對的模擬腳本。如下圖範例所示



此部分可參考 SUMO 的網頁:

http://sourceforge.net/apps/mediawiki/sumo/index.php?title=AdditionalTools_Trip#ra

ndomTrips.py

Appendix III: TraCI 的使用與範例(進階用法)

有關TraCI的資訊，請參考SUMO的官方網站:

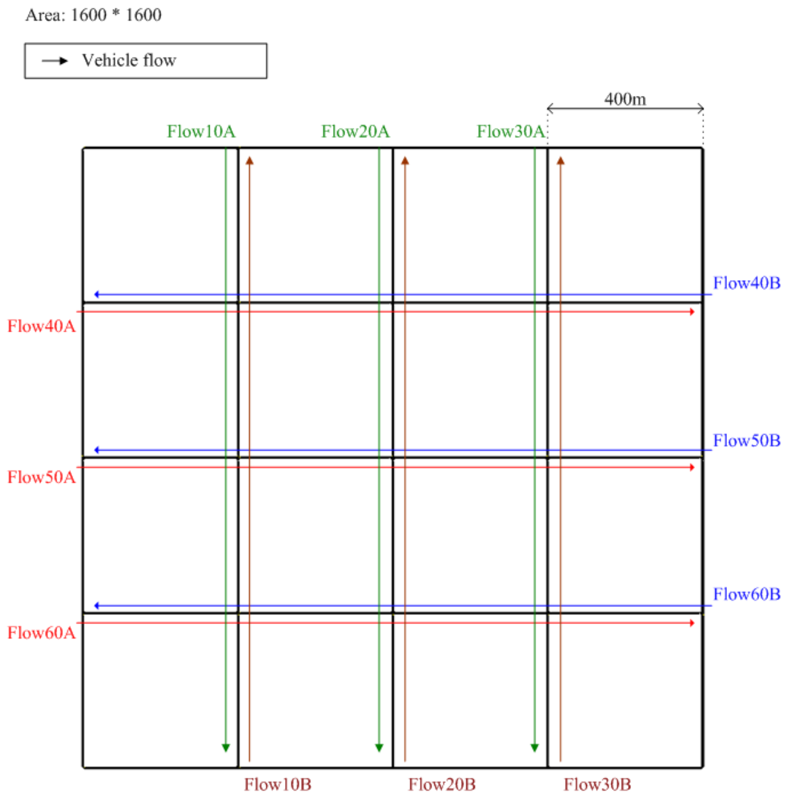
http://sourceforge.net/apps/mediawiki/sumo/index.php?title=TraCI

該範例使用SUMO的相關工具來實現。

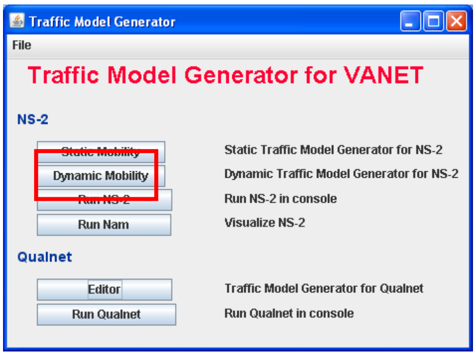
在此僅提供一些以MOVE於TraCI的範例，請使用我們的範例檔(for trace-version)

http://lens.csie.ncku.edu.tw/MOVE/TraCI_package_by_CMC.rar

來表現此案例，此案例使用的是一個方格狀的地圖，如以下所示。

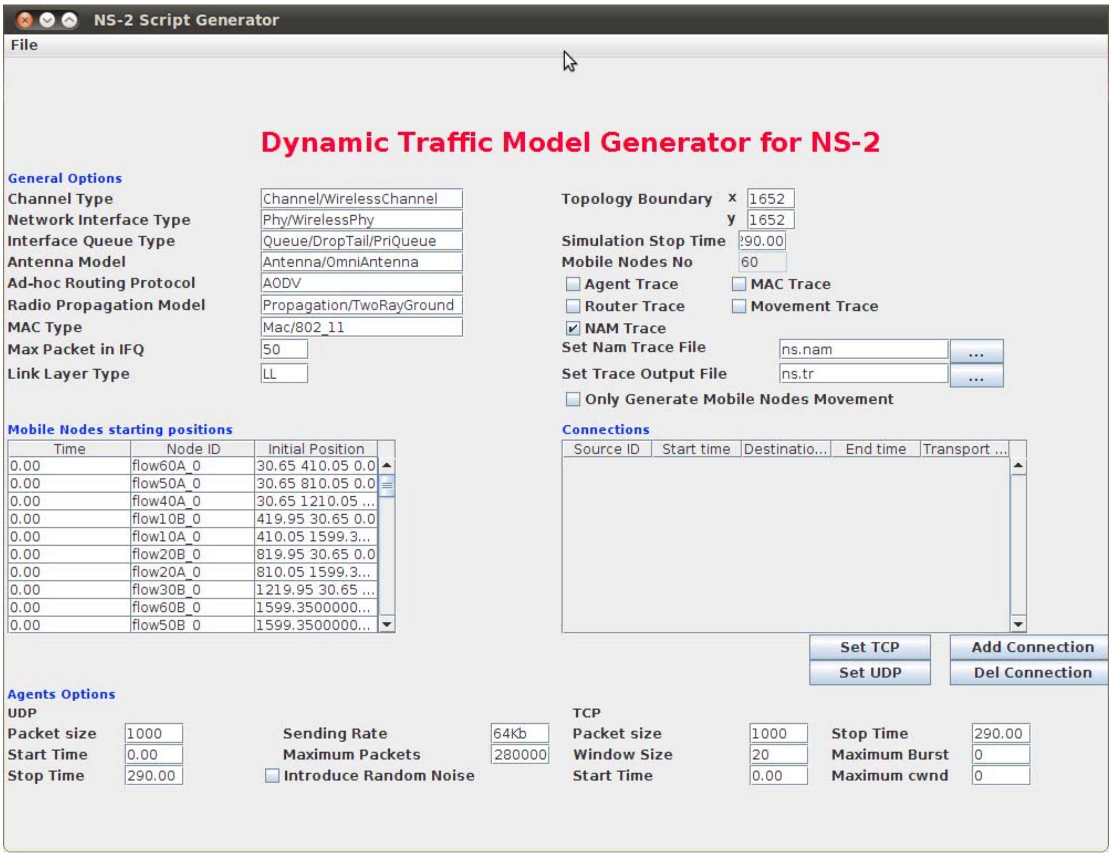


步驟一：請選擇 “dynamic mobility”



注意,此GUI只會產生相容於使用TraCI的TCL檔

步驟二：匯入地圖檔以及路線檔。

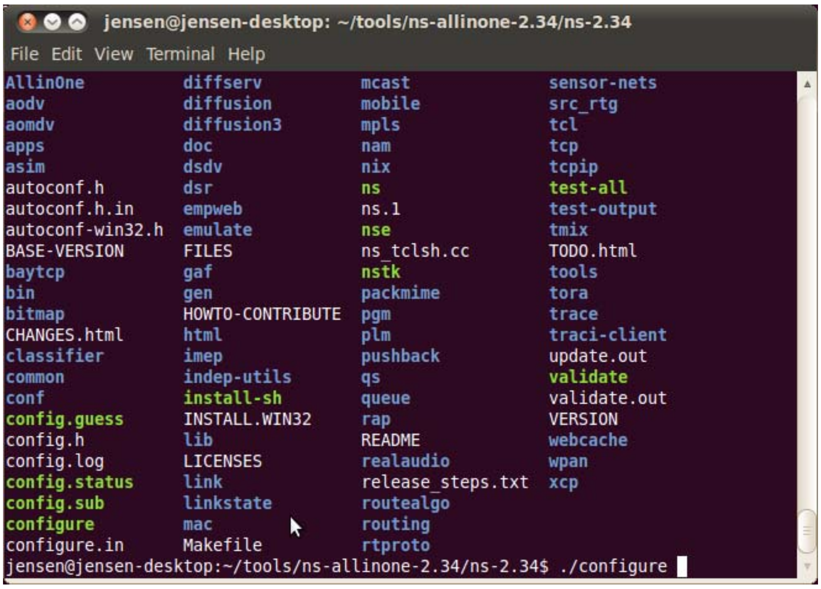


此範例中，我們將檔名訂為 “ex_traci.tcl”.

在開始模擬之前，我們要先安裝ns2 trace-patch.

直接將所有檔案複製到ns-2資料夾 (e.g. tcpip, traci-client, and Makefile.in).

在 ns-2 資料夾輸入 “./configure”



接著輸入 “make”

在此範例中，我們將示範一些TraCI API，如下所示:

1. command-GetRoadID [nsID] [sumoID]

Returns the id of the edge the named vehicle was at within the last step; error value: ""

$ns_ at 3.0 "$mobilityInterfaceClient command-GetRoadID 6 flow20A_0"

2. command-SetMaxSpeed [nsID] [sumoID] [max speed]

Sets the vehicle's maximum speed to the given value

$ns_ at 5.0 "$mobilityInterfaceClient command-SetMaxSpeed 0 flow60A_0 5"

3. command- changeTarget [nsID] [sumoID] [target edge]

The vehicle's destination edge is set to the given. The route is rebuilt.

$ns_ at 5.0 "$mobilityInterfaceClient command-ChangeTarget 0 flow60A_0 4344-2"

4. command-changeRoute [nsID] [sumoID] [number of edges] [edges lists]

Assigns the list of edges as the vehicle's new route assuming the first edge given is the one the vehicle is curently at: The first occurence of the edge is currently at is searched within the new route; the vehicle continues the route from this point in the route from. If the edge the vehicle is currently does not exist within the new route, an error is generated.

$ns_ at 5.0 "$mobilityInterfaceClient command-changeRoute 0 flow60A_0 7 0111-1,1011-2,1020-1,2021-1,2131-1,3141-1,3141-2"

步驟三: 執行 TraCI 伺服器 (重要)

在執行ns-2 模擬器之前，你每次都需要啟動 TraCI 伺服器。

此指令格式如下

Sumo –n [your map file] –r [your route file] --remote-port 8888 --penetration 1 --route-steps -1

範例如下:

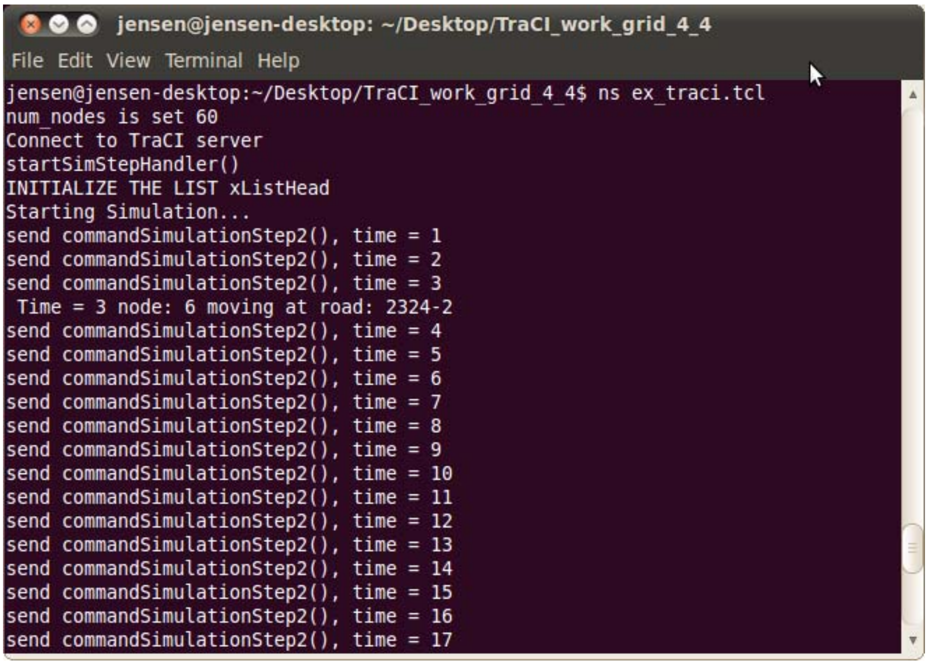
Sumo –n /yourPath/TraCI_work_grid_4_4/grid.net.xml -r /yourPath /TraCI_work_grid_4_4/grid.rou.xml --remote-port 8888 --penetration 1 --route-steps -1

步驟四：測試指令-GetRoadID

請將“ex_traci.tcl”中的註解移除(i.e. #)。

$ns_ at 3.0 "$mobilityInterfaceClient command-GetRoadID 6 flow20A_0"

你將得到道路ID“2324-2”如下圖所示

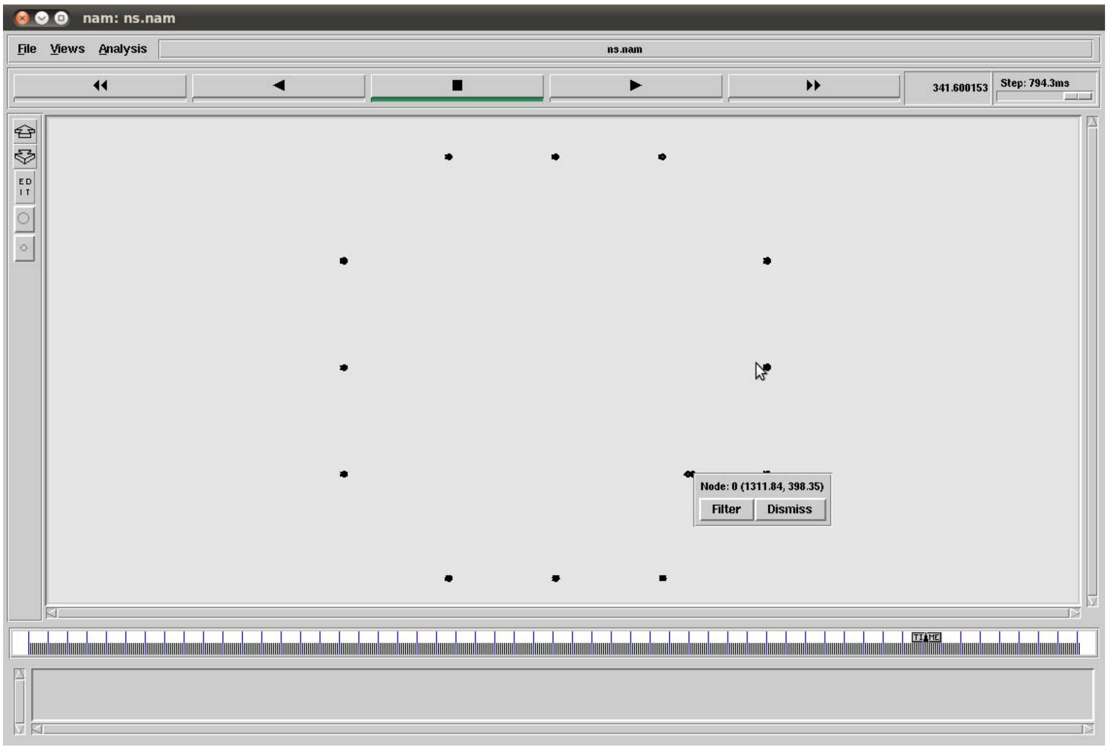


步驟五：測試指令-SetMaxSpeed

請將“ex_traci.tcl”中的註解移除(i.e. #)。

$ns_ at 5.0 "$mobilityInterfaceClient command-SetMaxSpeed 0 flow60A_0 5"

一旦你改變了車輛的速度，此節點將以特定的速度移動直到你再改變它為止。此結果如下圖所示，因為Node 0的最大速度為 5 公尺/秒，所以移動的較慢。

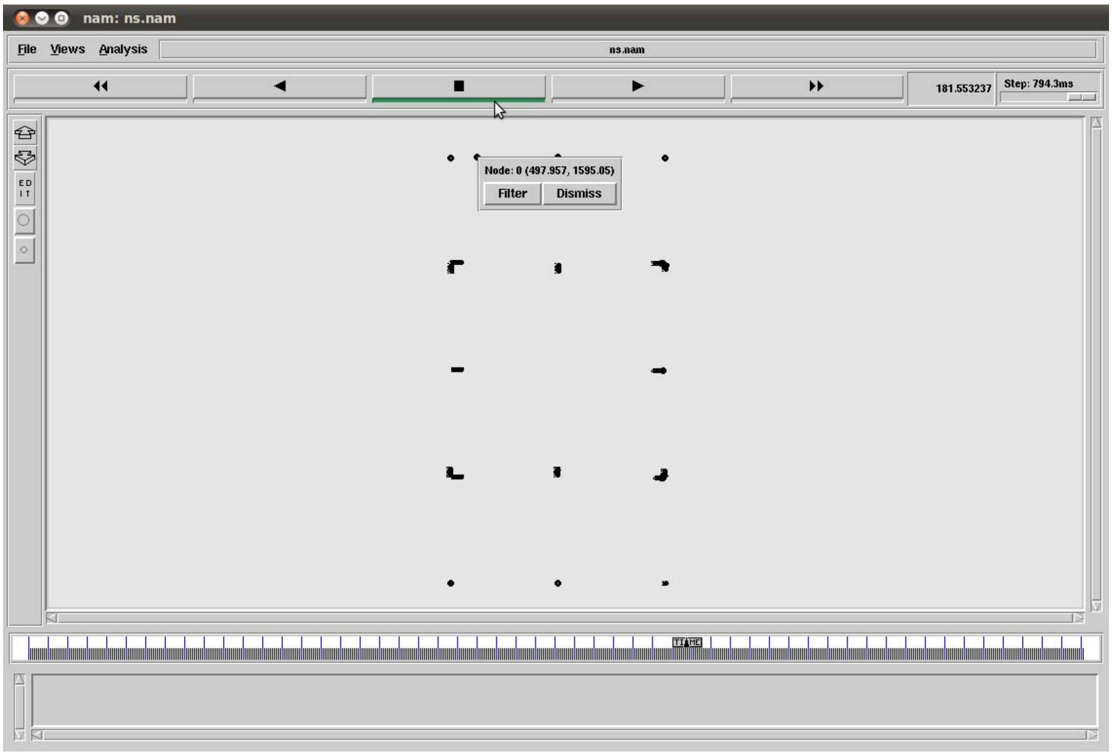


步驟六：測試指令-ChangeTarget

請將“ex_traci.tcl”中的註解移除(i.e. #)。

$ns_ at 5.0 "$mobilityInterfaceClient command-ChangeTarget 0 flow60A_0 4344-2"

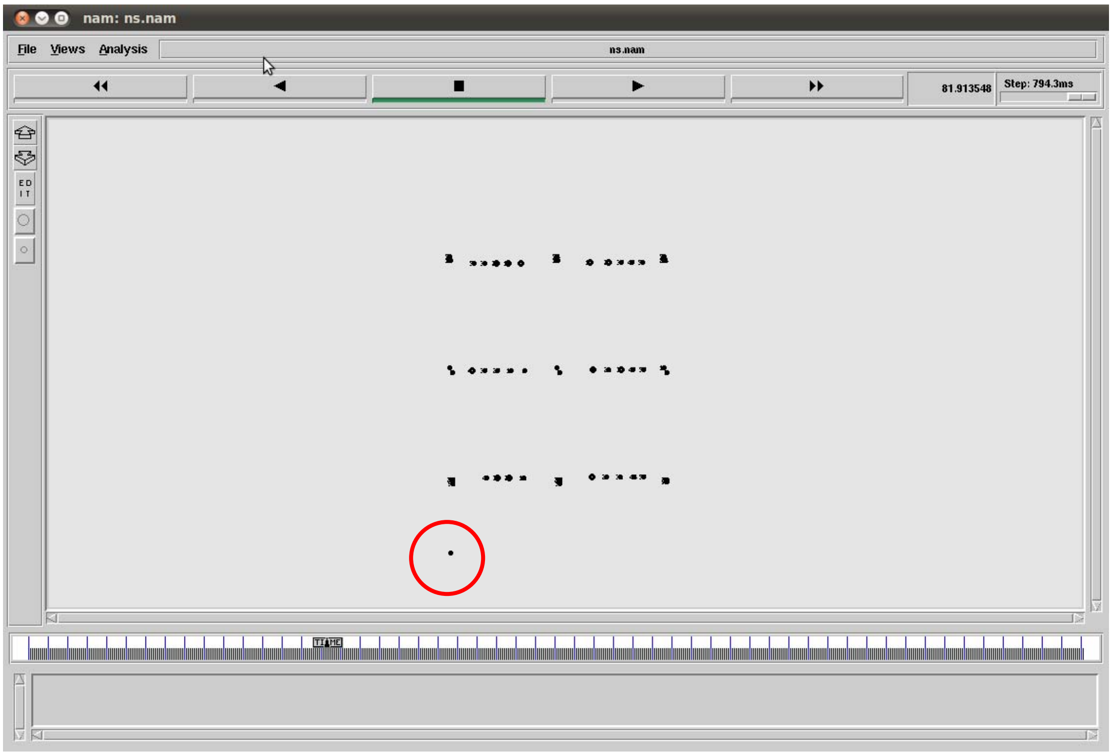
當你使用了此指令，節點將改變路線為最短路徑(SUMO將自動計算)，在此範例中，Node 0 (i.e. flow60A_0)將改變路徑到目標道路”4344-2”。



步驟七：測試指令-changeRoute

請將“ex_traci.tcl”中的註解移除(i.e. #)。

$ns_ at 5.0 "$mobilityInterfaceClient command-changeRoute 0 flow60A_0 7 0111-1,1011-2,1020-1,2021-1,2131-1,3141-1,3141-2"

在此指令中，你必須輸入車輛的當下道路為第一道路(i.e.0111-1)，再由當下道路到目的道路，輸入每一道路。舉例來說0111-1, 1011-2, 1020-1, 2021-1, 2131-1, 3141-1, 3141-2, 在此0111-1為當下道路3141-2 為目的地道路。

此結果如下圖所示，Node 0行進的路線與其他Node不同(即紅色圓圈)